Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)

Кафедра Автоматизації проектування обчислювальної техніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Методи захисту важливих даних у медичній IoT системі для розробки ліків
(тема)

Виконав:
здобувач 2 року навчання,
групи СКСм-24-1

Коровіна Д.С.
(прізвище, ініціали)

Спеціальність *123 Комп'ютерна інженерія*
(код і повна назва спеціальності)

Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Спеціалізовані комп'ютерні системи*
(повна назва освітньої програми)

Керівник асистент Хаханов І.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Чумаченко С. В
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Безпеки інформаційних технологій

Рівень вищої освіти другий (магістерський)

Спеціальність 123 Комп'ютерна інженерія
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Спеціалізовані комп'ютерні системи
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Коровіній Дар'ї Сергіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Методи захисту важливих даних у медичній IoT системі для розробки ліків

затверджена наказом університету від 07 листопада 2025 р. № 1012Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 13 01 2026 р.

3. Вихідні дані до роботи:

1. Теоретичний матеріал щодо життєвого циклу розробки захищених медичних систем
2. Теоретичний матеріал щодо управління технічними обмеженнями при впровадженні криптографічних методів
3. Теоретичний матеріал щодо вимог міжнародних стандартів щодо захисту персональних даних

4. Перелік питань, що потрібно опрацювати в роботі:

1. Аналіз предметної області.
2. Аналіз загроз інформаційній безпеці в системах IoT, класифікація критичних даних
3. Оцінка впливу захищених протоколів на енергоефективність та швидкість
4. Формування стратегії захисту

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) презентаційний матеріал у вигляді 14 слайдів

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

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк виконання етапів роботи	Примітка
1	Аналіз теми.	02.09.25 — 06.09.25	
2	Підготовка плану та структури роботи.	07.09.25 — 14.09.25	
3	Огляд літератури та збір теоретичних матеріалів.	15.09.25 — 01.10.25	
4	Розробка моделі та підходів до збору даних.	02.10.25 — 12.10.25	
5	Проведення експерименту. Обробка даних.	13.10.25 — 20.10.25	
6	Аналіз результатів і формулювання висновків.	21.10.25 — 25.10.25	
7	Написання основних розділів роботи.	26.10.25 — 24.11.25	
8	Оформлення рисунків, таблиць та посилань.	25.11.25 — 13.12.25	
9	Оформлення та редагування згідно вимог.	14.12.25 — 21.12.25	
10	Захист	22.12.25 — 24.12.25	

Дата видачі завдання 07 листопада 2025 р.

Здобувач _____ Коровіна Д.С.
(підпис)

Керівник роботи _____ Хаханов І.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 63 сторінок, 11 рисунків, 20 джерел, 3 таблиqs.

IOT, РОЗУМНИЙ ДОЗАТОР ЛІКІВ, МЕДИЧНІ ДАНІ, КІБЕРБЕЗПЕКА, ШИФРУВАННЯ, ПРИВАТНІСТЬ ДАНИХ

Об’єкт дослідження – процеси управління розробкою та впровадженням інтелектуальних медичних IoT-систем із підвищеними вимогами до захисту персональних даних.

Мета дослідження – обґрунтувати та реалізувати методи захисту конфіденційної інформації (графіків та історії прийому ліків) у медичній IoT-системі, враховуючи обмежені обчислювальні ресурси мікроконтролерів та специфіку управління медичними проєктами.

Актуальність. Стрімкий розвиток «Інтернету медичних речей» (IoMT) дозволяє автоматизувати контроль за станом пацієнтів, проте створює серйозні ризики для приватності. Для систем контролю прийому ліків витік даних або підміна команд управління пристроєм можуть мати критичні наслідки для здоров'я. Проблема ускладнюється тим, що малопотужні пристрої (ESP32/ESP8266) не завжди здатні підтримувати складні промислові стандарти шифрування. Неефективне управління таким проєктом призводить до вразливостей у каналах зв'язку (MQTT, HTTP) та базах даних. Тому розробка та вибір методологій, що дозволяють інтегрувати надійні методи захисту (AES-шифрування, TLS-протоколи) у легковагові IoT-рішення, є критично важливим завданням для сучасної медичної інженерії.

ABSTRACT

The explanatory note consists of 63 pages, 11 figures, 20 sources, and 3 tables.

Keywords: IOT, SMART MEDICINE DISPENSER, MEDICAL DATA, CYBERSECURITY, ENCRYPTION, DATA PRIVACY.

Object of study: Processes of managing the development and implementation of intelligent medical IoT systems with enhanced requirements for personal data protection.

Purpose of the study: To justify and implement methods for protecting confidential information (medication schedules and intake history) within a medical IoT system, considering the limited computational resources of microcontrollers and the specifics of medical project management.

Relevance: The rapid development of the "Internet of Medical Things" (IoMT) enables automated patient monitoring but poses significant privacy risks. For medication control systems, data leaks or tampering with device control commands can have critical health consequences. The problem is exacerbated by the fact that low-power devices (ESP32/ESP8266) are not always capable of supporting complex industrial encryption standards. Ineffective management of such projects leads to vulnerabilities in communication channels (MQTT, HTTP) and databases. Therefore, the development and selection of methodologies that integrate reliable protection methods (AES encryption, TLS protocols) into lightweight IoT solutions is a critically important task for modern medical engineering.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, ОДИНИЦЬ І ТЕРМІНІВ

ACID – Atomicity, Consistency, Isolation, Durability

AES – Advanced Encryption Standard

API – Application Programming Interface

CA – Certificate Authority

CBC – Cipher Block Chaining

ERD – Entity-Relationship Diagram

GDPR – General Data Protection Regulation

GPIO – General Purpose Input/Output

HIPAA – Health Insurance Portability and Accountability Act

HTTPS – HyperText Transfer Protocol Secure

IoMT – Internet of Medical Things

IoT – Internet of Things

IV – Initialization Vector

JSON – JavaScript Object Notation

JWT – JSON Web Token

MPR – Medication Possession Ratio

MQTT – Message Queuing Telemetry Transport

NTP – Network Time Protocol

ORM – Object-Relational Mapping

PI – Punctuality Index

PKI – Public Key Infrastructure

REST – Representational State Transfer

SDLC – Software Development Life Cycle

SMTP – Simple Mail Transfer Protocol

TLS – Transport Layer Security

UI/UX – User Interface / User Experience

ВСТУП

У сучасному цифровому середовищі технології Інтернету речей (IoT) стали невід'ємною частиною повсякденного життя, відкриваючи нові горизонти для автоматизації та оптимізації процесів у різних сферах, включаючи охорону здоров'я [1]. Поширення IoT-пристроїв, зокрема, таких як розумні дозатори ліків, забезпечило користувачам безпрецедентну зручність у моніторингу та управлінні прийомом медикаментів, від персоналізованих нагадувань до віддаленого контролю. Однак, це широке впровадження також викликає значні проблеми з безпекою, зокрема щодо захисту чутливих медичних даних, які збираються, обробляються та передаються цими пристроями.

IoT-системи в медицині, включаючи розумні дозатори ліків, працюють з надзвичайно конфіденційною інформацією про стан здоров'я користувачів, їхні медичні призначення та історію прийому ліків. Ці дані є привабливою мішенню для кібератак, витоків інформації та несанкціонованого доступу. Кібербезпека в медичних IoT-пристроях є критично важливою, оскільки компрометація таких даних може призвести не лише до порушення приватності, а й до серйозних наслідків для здоров'я пацієнтів через маніпуляції з інформацією про лікування. Типові автономні пристрої IoT зазвичай мають обмежену ємність та низьку потужність обробки, що ускладнює впровадження складних механізмів криптографічного захисту.

Проблема захисту чутливих медичних даних у системах IoT не є просто теоретичною. Було задокументовано численні випадки, коли вразливості в медичних пристроях або їхньому програмному забезпеченні призводили до витоків даних або можливостей для зловмисного втручання. Це включає несанкціонований доступ до медичних записів, зміну даних про дозування, відстеження переміщень користувачів або навіть виведення з ладу медичних пристроїв. Такі інциденти підкреслюють нагальну необхідність ретельного

дослідження та впровадження надійних методів захисту медичних даних у системах IoT для мінімізації потенційних ризиків безпеці.

Навіть при тому, що розробники прагнуть створювати безпечні IoT-рішення, а медичні регулятори вводять стандарти захисту даних (наприклад, HIPAA, GDPR), зловмисники постійно шукають нові шляхи для обходу існуючих заходів безпеки. Обмежені обчислювальні ресурси та енергоспоживання багатьох IoT-пристроїв створюють додаткові виклики для реалізації ефективних криптографічних протоколів та механізмів автентифікації.

Основною метою цієї роботи є дослідження методів захисту чутливих медичних даних у системі IoT для розумного дозатора ліків. Це передбачає не лише вибір та обґрунтування відповідних криптографічних протоколів і механізмів автентифікації, а й розуміння контексту, у якому ці дані використовуються, а також інтеграцію сучасних підходів, таких як машинне навчання, для підвищення ефективності системи. Проект спрямований на створення розумного дозатора ліків, який фіксує час прийому медикаментів та взаємодіє з мобільним додатком. Останній буде нагадувати користувачам про необхідність прийому ліків, сповіщати опікунів у разі пропущеної дози, використовувати алгоритми машинного навчання для виявлення патернів у поведінці користувача та прогнозування пропуску прийому ліків для адаптивного сповіщення.

Хоча функціональність розумного дозатора очевидна, кінцевому користувачеві може бути незрозуміло, як саме забезпечується захист його чутливої інформації. Наприклад, система дозатора потребує доступу до даних про розклад прийому ліків та історії їх вживання для забезпечення нагадувань. У цьому випадку користувач має бути впевненим, що програма використовує цю конфіденційну інформацію лише за призначенням і не передає її третім сторонам без згоди. Комплексний аналіз методів захисту дозволить відрізнити системи, що лише декларують безпеку, від тих, що реально впроваджують її на всіх рівнях взаємодії.

Для підтвердження ефективності запропонованих методів захисту буде детально проаналізовано застосовність різних криптографічних рішень та протоколів безпеки до архітектури розумного дозатора. Цей аналіз висвітлить загальні тенденції щодо захисту медичних даних в IoT та виявить конкретні рішення, які демонструють високий рівень безпеки та конфіденційності.

Підсумовуючи, цей проект спрямований на вирішення критично важливого аспекту медичної IoT-безпеки шляхом дослідження та впровадження надійних методів захисту чутливих медичних даних у системі розумного дозатора ліків. Ця робота принесе користь не лише окремим користувачам, підвищуючи їхню довіру до медичних IoT-рішень, але й зробить вагомий внесок в академічний і професійний дискурс щодо безпеки медичних пристроїв, закладаючи основу для майбутніх досліджень і розробок у цій життєво важливій галузі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 IoT у медицині: нова ера цифрового здоров’я

Ефективне дотримання режиму прийому медикаментів є критично важливим для успіху лікування багатьох захворювань, особливо хронічних. Проте, значний відсоток пацієнтів регулярно пропускає прийом ліків або вживає їх неправильно, що призводить до погіршення стану здоров'я, госпіталізацій та зростання витрат на охорону здоров'я. Сучасні технології, зокрема Інтернет речей (IoT), пропонують нові можливості для вирішення цієї проблеми шляхом автоматизації процесу моніторингу та нагадувань про прийом ліків. Водночас, використання чутливих медичних даних у таких системах висуває суворі вимоги до їхньої безпеки та конфіденційності.

Internet of Things (IoT), або Інтернет речей, – це концепція, яка передбачає взаємодію фізичних пристроїв з підключенням до мережі для збору, передачі й обробки даних. У медицині ця технологія дала початок підгалузі Internet of Medical Things (IoMT), яка об'єднує інтелектуальні пристрої для моніторингу здоров’я, терапевтичної підтримки та телемедицини [2].

Переваги використання IoT у медицині включають:

- постійний моніторинг пацієнта в реальному часі (наприклад, кардіостимулятори, пульсометри, глюкометри);
- зменшення навантаження на медичний персонал завдяки автоматизації обліку та контролю;
- підвищення точності та своєчасності медичних рішень;
- індивідуалізація лікування за рахунок аналізу даних про поведінку пацієнта;
- підвищення прихильності до призначеної терапії (adherence), особливо при хронічних захворюваннях.

Пристрої такого типу вже використовуються у пульсоксиметрії,

кардіології, ендокринології (наприклад, розумні інсулінові помпи), а також у системах реабілітації та догляду за літніми людьми.

1.2 Вразливість IoT-медичних систем: виклики безпеки

Інтеграція IoT-технологій у сферу охорони здоров’я принесла значні переваги у вигляді автоматизації моніторингу стану пацієнтів, покращення точності збору медичних даних і забезпечення віддаленої діагностики. Проте ці ж самі технології створили нову поверхню для атак, оскільки в більшості випадків IoT-пристрої проєктувалися без глибокого врахування вимог до безпеки. Вразливості IoT-систем обумовлені насамперед низькою обчислювальною потужністю, нестабільними мережевими з'єднаннями та слабким програмним захистом, що у сукупності створює ідеальне середовище для зловмисників [3].

Однією з ключових проблем є апаратні обмеження типових IoT-плат. Пристрої, які використовуються для моніторингу здоров’я (наприклад, розумні дозатори ліків), базуються переважно на мікроконтролерах із 8- або 32-бітною архітектурою, що мають мінімальний обсяг оперативної пам’яті, невелику кількість енергії, обмежений набір портів і відсутність вбудованого шифрування. У таких умовах традиційні алгоритми безпеки, як-от RSA з довгими ключами чи повноцінний TLS-протокол, є надто ресурсоємними. Впровадження стандартних криптографічних методів або неможливе, або викликає затримки в роботі пристрою, що неприпустимо в системах, де час реагування критичний, наприклад, при нагадуванні про прийом життєво важливих препаратів.

Ще одним суттєвим ризиком є використання нестабільних або відкритих каналів зв’язку. Значна частина IoT-пристроїв передає дані через бездротові технології типу Wi-Fi, Bluetooth Low Energy (BLE) або ZigBee. Усі ці протоколи мають відомі вразливості, які можуть бути використані для атак типу «людина посередині» (MITM), перехоплення, модифікації або підміни

переданих даних. Наприклад, у багатьох комерційних медичних гаджетах аутентифікація реалізована на базовому рівні або взагалі відсутня, що дозволяє третім особам здійснювати підключення до пристрою, не маючи відповідних прав доступу. У найгіршому випадку це може призвести до навмисної зміни графіка прийому ліків або повної дезактивації системи нагадування.

Ще однією критичною загрозою є взаємодія з хмарними сервісами. Сучасні IoT-рішення в медицині часто передбачають зберігання даних пацієнтів на сторонніх серверах або у хмарі. Якщо обробка даних у хмарі не супроводжується належним шифруванням, контролем доступу та журналюванням подій, це створює додаткові точки вразливості. Сервери можуть бути зламані, а дані – скомпрометовані, що ставить під загрозу не лише конфіденційність пацієнтів, а й репутацію та юридичну відповідальність медичних установ. Крім того, в умовах регламентів на зразок GDPR або HIPAA, витік навіть мінімальної кількості персональних даних може мати серйозні правові наслідки.

Слід зазначити, що оновлення програмного забезпечення для IoT-пристроїв часто відсутнє або реалізоване неналежним чином. Багато пристроїв не підтримують механізмів захищеного віддаленого оновлення (firmware over-the-air, OTA), що унеможливлює виправлення вразливостей після їх виявлення. Це означає, що з часом рівень безпеки IoT-пристрою лише знижується, і він стає дедалі привабливішою ціллю для атак. У деяких випадках навіть відомі вразливості залишаються не виправленими протягом довгого часу, оскільки виробник не має технічної або фінансової можливості підтримувати пристрій у довгостроковій перспективі.

Окрім технічних загроз, існує ще й соціальна складова проблеми. Користувачі, особливо літні люди або пацієнти з когнітивними порушеннями, часто не розуміють ризиків, пов’язаних із використанням підключених пристроїв. Вони можуть залишити пристрій без нагляду, дозволити стороннім особам отримати доступ або не оновити додаток, що призводить до появи

нових точок вразливості. У такому середовищі будь-який захист має бути максимально автоматизованим, прозорим і ненав’язливим, що ще більше ускладнює реалізацію ефективних механізмів безпеки.

Таким чином, захист IoT-систем у медицині є складним багатогранним завданням. Воно включає забезпечення криптографічної цілісності й конфіденційності переданих даних, автентифікацію пристроїв і користувачів, контроль доступу до хмарних сервісів, захищене оновлення програмного забезпечення та врахування людського фактора. Без вирішення цих проблем масове впровадження IoT у медичну практику може мати більше ризиків, ніж переваг, особливо в контексті захисту персональних даних пацієнтів.

1.3 Системи моніторингу прийому ліків: проблема прихильності до лікування

Важливою проблемою в лікуванні пацієнтів із хронічними захворюваннями (гіпертонія, діабет, епілепсія тощо) є несистематичний прийом ліків або повне ігнорування графіка дозування. За даними WHO, до 50% пацієнтів не дотримуються приписаного режиму терапії, що призводить до загострення стану, рецидивів та ускладнень. Розумні дозатори ліків вирішують цю проблему завдяки механічному контролю видачі (ліки не можна взяти самовільно або випадково пропустити), фіксації часу прийому, нагадуванням користувачам про необхідність прийому (світлові/звукові сигнали, повідомлення), сповіщенням родичів або лікаря, якщо доза пропущена.

Такі системи мають найвищу цінність для пацієнтів з розладами пам’яті (наприклад, деменція), дітей і літніх людей, осіб, що проходять тривале лікування, яке потребує суворої схеми прийому.

1.4 Огляд сучасних IoT-систем для моніторингу прийому ліків: Аналіз існуючих рішень на ринку, їх переваги та недоліки.

Ринок медичних IoT-пристроїв активно розвивається, пропонуючи різноманітні рішення для підвищення дотримання медикаментозних рекомендації лікарів. Існуючі системи можна класифікувати за їхньою складністю, функціоналом та цільовою аудиторією.

Hero Health (рисунок 1.1) – це автоматичний дозатор ліків, який зберігає до 10 різних медикаментів, автоматично видає потрібні дози за розкладом, надає звукові та візуальні нагадування. Має підключення до Wi-Fi та мобільний додаток для моніторингу та сповіщень для опікунів. З переваг можна зазначити високу автоматизацію, зручність, але сам пристрій має високу вартість, потенційні вразливості в кібербезпеці, що типові для складних IoT-пристроїв.

Рисунок 1.1 – Автоматичний дозатор ліків Hero Health

Philips Medication Dispensing Service (рисунок 1.2) – це автоматичний дозатор, який надає нагадування та можливість дистанційного моніторингу. Орієнтований переважно на людей похилого віку. Використовувати його просто, а також він надійний. З іншого боку, має обмежений функціонал та відсутність адаптивних алгоритмів.

Рисунок 1.2 – Автоматичний дозатор ліків Philips

Таким чином, виділяємо переваги, недоліки існуючих рішень. Переваги складаються з:

- зменшення навантаження на опікунів;
- автоматизація рутинних процесів;
- можливість збору даних для подальшого аналізу.

Недоліки:

- проблеми з кібербезпекою – більшість комерційних рішень фокусуються на функціоналі та зручності, залишаючи питання надійного захисту медичних даних часто недооціненими або реалізованими недостатньо, обмежені обчислювальні ресурси та енергоспоживання багатьох IoT-

пристроїв ускладнюють впровадження криптографічних протоколів;

- відсутність адаптивності – багато систем надають лише статичні нагадування, не враховуючи індивідуальні особливості поведінки пацієнтів та не прогнозуючи можливі пропуски;

- висока вартість – деякі комерційні рішення є дорогими та недоступними для широкого кола споживачів;

- залежність від інфраструктури – покладаються на постійне інтернет-з'єднання або стільниковий зв'язок;

- складність налаштування – деякі системи можуть бути занадто складними для використання людьми похилого віку або з обмеженими технологічними навичками.

2 ТЕОРЕТИЧНІ ОСНОВИ ТА ОГЛЯД МЕТОДІВ ЗАХИСТУ ДАНИХ В IOT

2.1 Класифікація чутливих медичних даних та вимоги до їх захисту

Ефективний захист даних у системах Інтернету речей (IoT), особливо тих, що оперують надзвичайно чутливою інформацією, вимагає глибокого та всебічного розуміння основних концепцій кібербезпеки [4]. В архітектурі IoT-систем, яка зазвичай включає сенсорні пристрої, мережевий шар та серверну або хмарну інфраструктуру, кожен з цих рівнів представляє унікальні вразливості до різноманітних кібератак. Зокрема, можливе перехоплення мережевого трафіку, як секретного, так і відкритого, атаки типу "людина-посередині" (Man-in-the-Middle, MITM), а також фальсифікація чи модифікація даних. Забезпечення базових принципів інформаційної безпеки – конфіденційності, цілісності та доступності (тріада CIA) – є основоположним. Це передбачає обов'язкове шифрування даних у каналі зв'язку, надійну автентифікацію пристроїв та строгий контроль доступу. У контексті медичних систем, крім загальних вимог безпеки, необхідно враховувати й жорстке законодавче регулювання. Наприклад, у Сполучених Штатах Америки діє Закон про переносність та підзвітність медичного страхування (HIPAA), а в Європейському Союзі – Загальний регламент про захист даних (GDPR), які встановлюють суворі вимоги до захисту персональної медичної інформації та передбачають значні штрафи за їх недотримання[13, 14].

Чутливі медичні дані, часто позначаються як Захищена медична інформація (Protected Health Information – PHI), становлять одну з найбільш критичних категорій даних, що вимагає найвищого рівня захисту. Їхня компрометація несе в собі значні ризики, починаючи від фінансових збитків і закінчуючи серйозними наслідками для здоров'я пацієнтів, можливістю дискримінації та тотальним порушенням приватності.

У контексті функціонування розумного дозатора ліків та пов'язаної з ним IoT-системи, можна виділити наступні ключові категорії чутливих медичних даних. По-перше, це персональні ідентифіковані дані (PII), які прямо або опосередковано дозволяють ідентифікувати конкретну особу, як пацієнта, так і його опікуна. Сюди належать повні прізвище, ім'я та по батькові, дата народження, точний вік, а також всіляка контактна інформація, така як адреса електронної пошти, номери телефонів та фізична адреса проживання. Крім того, до PII відносяться унікальні системні ідентифікатори користувача, MAC-адреси пристроїв та IP-адреси, які, хоча й не є прямими ідентифікаторами особи, можуть бути використані для її деанонімізації при об'єднанні з іншими даними. Інформація про опікунів, включаючи їхні контактні дані та роль у системі моніторингу, також підпадає під цю категорію.

По-друге, це безпосередньо дані про здоров'я (Health Information), що охоплюють будь-яку інформацію, пов'язану з фізичним чи психічним станом здоров'я людини, наданими їй медичними послугами або ж фінансовими аспектами таких послуг. У даній системі до цієї категорії належать детальні медичні призначення, включаючи точні назви лікарських препаратів, необхідне дозування, частоту та визначений час прийому, а також загальну тривалість курсу лікування. Ключовим елементом є детальна історія прийому ліків, що включає точні дані про фактично прийняті та, що не менш важливо, пропущені дози, разом із точним часом кожної події, зафіксованої дозатором. Окрім цього, система збирає поведінкові патерни, які використовуються алгоритмами машинного навчання. Ці дані відображають регулярність прийому ліків, реакцію пацієнта на нагадування, типовий час доби, коли відбуваються пропуски, та інші індивідуальні особливості. Навіть якщо ці поведінкові дані піддаються анонімізації або псевдонімізації для аналізу, у своєму сирому вигляді вони можуть бути пов'язані з конкретним пацієнтом, що робить їх чутливими.

Вимоги до захисту чутливих медичних даних є надзвичайно строгими і

регламентуються низкою міжнародних та національних стандартів та законодавчих актів. Фундаментом цих вимог є триєдина модель інформаційної безпеки: конфіденційність, цілісність та доступність (тріада CIA).

Конфіденційність вимагає забезпечення того, що медична інформація доступна виключно авторизованим особам, об'єктам або процесам. У контексті медичних даних це означає, що лише пацієнт, його офіційний опікун (з попереднього дозволу пацієнта) та авторизований медичний персонал мають право доступу до цієї інформації. Для IoT-систем, досягнення конфіденційності реалізується через надійне шифрування даних як під час їх передачі через мережу, так і під час їх зберігання на пристроях та серверах. Це мінімізує ризик несанкціонованого перехоплення чи доступу.

Цілісність гарантує точність і повноту інформації, а також непорушність методів її обробки. Це означає, що дані не повинні бути змінені, пошкоджені або видалені несанкціонованим чином. Для системи є критично важливим, щоб дані про фактично прийняті дози не могли бути підроблені або спотворені, і щоб призначений розклад прийому ліків не міг бути несанкціоновано змінений зловмисником. Забезпечення цілісності досягається за допомогою застосування хешування, цифрових підписів та контрольних сум, які дозволяють виявити будь-які несанкціоновані зміни даних.

Доступність передбачає, що авторизовані користувачі мають постійний та своєчасний доступ до інформації та ресурсів системи. У випадку розумного дозатора, це означає, що пацієнт та його опікун повинні мати можливість переглядати розклад прийому ліків, історію прийомів, отримувати нагадування та сповіщення без перебоїв. Доступність також включає функціонування самого дозатора, який повинен видавати ліки за розкладом. Це забезпечується резервуванням систем, механізмами стійкості до відмов та захистом від атак типу "відмова в обслуговуванні" (DoS), які могли б перешкодити нормальній роботі системи.

Дотримання цих принципів є обов'язковим для будь-якої медичної IoT-системи, оскільки недотримання будь-якого з них може мати серйозні

наслідки для пацієнтів та медичних установ.

2.2 Вразливості протоколи MQTT та механізми атак

Сучасні системи медичного IoT, включаючи пристрої для видачі ліків, монітори життєво важливих показників та госпітальну автоматику, базуються на розподіленій архітектурі. Вона зазвичай включає три рівні: кінцеві сенсори або контролери (наприклад, на базі ESP32), мережеві шлюзи та централізовані хмарні сервери або мобільні додатки для управління та аналізу даних.

Протокол MQTT (Message Queuing Telemetry Transport) став де-факто стандартом для IoMT не лише через свою легкість, а й через здатність забезпечувати надійний зв'язок у специфічних медичних умовах. Його важливість виходить далеко за межі окремих апаратів видачі ліків і охоплює всю цифрову трансформацію охорони здоров'я [15].

MQTT дозволяє в реальному часі транслювати дані ЕКГ, рівень сатурації або температуру пацієнта безпосередньо лікарю, де пацієнт виступає "видавцем" даних, а медична консоль — "підписником".

Завдяки рівням якості обслуговування (QoS), MQTT забезпечує надійність, яка є життєво важливою для медицини. Зокрема, QoS 2 гарантує доставку повідомлення "точно один раз", що є критичним для фінансових транзакцій або команд на введення медикаментів, де втрата чи дублювання команди може мати фатальні наслідки.

Багато медичних сенсорів працюють від батарей. MQTT мінімізує використання мережевого ресурсу та енергії, дозволяючи пристроям тривалий час залишатися в режимі очікування, зберігаючи при цьому сесію зв'язку.

Протокол використовується для моніторингу умов зберігання ліків та автоматизації лабораторних досліджень, забезпечуючи цілісність даних на всіх етапах — від заводу до пацієнта.

Проте саме популярність та відкритість MQTT роблять його мішенню. Дослідження за допомогою Shodan показують тисячі публічно відкритих

MQTT-брокерів, які надають доступ не лише до домашньої автоматики, а й до кардіомоніторів, інсулінових помп та систем безпеки медичних установ.

Атаки на MQTT-мережі можна розділити на три фази: підключення, автентифікація та комунікація. На фазі підключення зловмисник може використовувати IP-спуфінг для перехоплення з'єднання. Під час фази автентифікації відсутність надійних механізмів перевірки дозволяє атакувальнику отримати несанкціонований доступ до топіків управління. Найбільш небезпечною є фаза комунікації, де стають можливими атаки типу Man-in-the-Middle (MitM) та підміна команд.

Атака на підміну команд (Command Spoofing) у системі видачі ліків має прямі загрози життю пацієнта. Зловмисник може опублікувати маніпульовану команду в топік, на який підписаний дозатор. Наприклад, команда на видачу однієї таблетки може бути змінена на видачу критичної дози. Використання сучасних методів, таких як BERT-моделі для генерації шкідливих повідомлень, дозволяє зловмисникам створювати атаки, які важко виявити традиційними системами захисту.

Таблиця 2.1 – Опис атак

Тип атаки	Механізм впливу на систему видачі ліків	Наслідки
MQTT Spoofing	Публікація фальшивих команд від імені лікаря	Передозування або пропуск прийому ліків
Replay Attack	Повторна відправка перехопленої команди на видачу	Подвійне дозування препарату
MITM Message Alteration	Зміна параметрів дозування "на льоту"	Непередбачуваний вплив на терапію
DoS (Denial of Service)	Перевантаження брок	Пристрій стає некерованим у критичний момент

Окрім фізичної небезпеки, існує загроза викрадення історії прийомів ліків або журналів вітальних показників. Оскільки MQTT дозволяє зберігати останні повідомлення (retained messages), новий підписник може миттєво отримати конфіденційну інформацію про стан здоров'я пацієнта, якщо права доступу налаштовані некоректно. Це порушує вимоги HIPAA та GDPR щодо захисту приватності медичних даних.

2.3 Апаратні обмеження ESP32 та криза стандартної криптографії

Мікроконтролери серії ESP32 є основою для багатьох пристроїв IoMT завдяки вбудованим інтерфейсам Wi-Fi та Bluetooth. Однак їхня архітектура накладає суворі обмеження на типи використовуваних криптографічних алгоритмів [7].

Платформа ESP32 представлена кількома варіантами, кожен з яких має свої особливості обробки даних. Основна модель базується на двоядерному процесорі Xtensa LX6 з частотою 240 МГц, тоді як нові версії, такі як ESP32-C3 та C6, використовують ядра RISC-V. Обсяг доступної SRAM зазвичай становить близько 520 КБ, з яких лише частина доступна для користувацьких додатків після завантаження стека Wi-Fi та операційної системи.

Таблиця 2.2

Варіант ESP32	Архітектура ядра	SRAM	Вбудовані апаратні прискорювачі
ESP32 (Original)	Dual-core Xtensa LX6	520 KB	AES, RSA, SHA, ECC (Limited)
ESP32-S3	Dual-core Xtensa LX7	512 KB	AES, RSA, SHA, ECC, AI Instructions
ESP32-C3	Single-core RISC-V	400 KB	AES, RSA, SHA, ECC (Full)

ESP32-C6	Single-core RISC-V	512 KB	AES, RSA, SHA, ECC, Wi-Fi 6 Support

Алгоритм RSA (Rivest-Shamir-Adleman) довгий час був стандартом для асиметричного шифрування. Його безпека базується на складності факторизації великих цілих чисел. Однак для забезпечення сучасного рівня безпеки необхідний ключ довжиною 3072 або 4096 біт.

Використання RSA на ESP32 супроводжується певними проблемами.

По-перше, операції з великими числами вимагають значних блоків у RAM. Для 4096-бітного RSA це може призвести до критичного браку пам'яті для інших функцій пристрою. По-друге, навіть з апаратною підтримкою, встановлення TLS-з’єднання за допомогою RSA на ESP32 займає відчутний час (близько 1.6 секунд), що створює затримки в реагуванні медичного обладнання. І наостанок – енерговитрати, процес модульного піднесення до степеня є обчислювально інтенсивним, що призводить до швидкого розряду батареї автономних пристроїв.

З огляду на ці фактори, RSA стає дедалі менш придатним для сучасних екосистем IoMT, і постає питання у пошуку нового, альтернативноо підходу.

2.4 Перехід до еліптичної криптографії (ECC) як рішення для IoMT

Еліптична криптографія (ECC) пропонує альтернативний підхід до асиметричного шифрування, що базується на математичній складності задачі дискретного логарифму в групі точок на еліптичній кривій [20].

Головна перевага ECC полягає в тому, що вона забезпечує аналогічний рівень безпеки при значно менших розмірах ключів. Наприклад, рівень безпеки 3072-бітного RSA досягається за допомогою лише 256-бітного ключа ECC.

На ESP32 використання ECC дозволяє значно прискорити процес цифрового підпису та генерації ключів. ECC домінує у швидкості створення

підпису, що є критичним для кінцевого IoT-пристрою, який відправляє медичні дані.

Сучасні версії ESP32, такі як серія C3 та S3, включають спеціалізовані апаратні блоки для прискорення ECC. Це дозволяє виконувати криптографічні операції з мінімальним залученням основного ядра процесора. Впровадження ECC також зменшує обсяг мережевого трафіку під час TLS-рукостискання, оскільки розмір сертифікатів стає значно меншим.

Для медичних систем, де критичними є швидкість передачі тривожних сигналів та цілісність даних (незмінність показників пацієнта), ECC є безальтернативним рішенням. Вона забезпечує "військовий" рівень захисту без перегріву пристрою та без надмірного споживання пам'яті (RAM/Flash), що робить архітектуру IoMT надійною та масштабованою.

2.5 Стратегії забезпечення безпеки та приватності в екосистемі IoMT на базі ESP32

Так як ми вже визначили, що традиційні методи шифрування стають неефективними, то виходом із цієї ситуації є впровадження легкої криптографії (Lightweight Cryptography, LWC). Ця спеціалізована галузь спрямована на створення алгоритмів, що забезпечують високий рівень захисту при мінімальних витратах енергії. Ключовим етапом розвитку LWC став конкурс NIST, за результатами якого у 2023 році основним стандартом було обрано родину алгоритмів Ascon [9]. Завдяки використанню конструкції «губка» (sponge construction) та оперуванню 320-бітним станом, Ascon реалізує механізм автентифікованого шифрування з асоційованими даними (AEAD). Це дозволяє пристрою IoMT одночасно гарантувати конфіденційність медичних показників та їхню цілісність, що є критичним для запобігання підміні даних про стан пацієнта. У порівнянні з іншими фіналістами, такими як SPECK (оптимізований для швидкості на ARX-архітектурах) або Xoodyak, Ascon демонструє найкращий баланс між

енергоефективністю та стійкістю до атак по сторонніх каналах саме на 32-бітних архітектурах, як-от ESP32.

Поряд із новітніми стандартами LWC, у практичній реалізації IoMT-систем постає питання вибору між перевіреними симетричними алгоритмами: AES-128 та ChaCha20. Для розробників на платформі ESP32 цей вибір часто визначається наявністю апаратного прискорення. AES-128 залишається "золотим стандартом" завдяки вбудованим у чіпи ESP32 (серії S3, C3) апаратним акселераторам, які дозволяють виконувати режим AES-GCM з мінімальним залученням центрального процесора. Це забезпечує найвищу пропускну здатність, проте вимагає обережності при програмній реалізації через ризик атак за часом на таблиці підстановок (S-boxes). Як безпечну альтернативу розглядають потоковий шифр ChaCha20. Оскільки він базується на простих операціях додавання, циклічного зсуву та XOR (ARX), він виконується за фіксований час навіть у програмному варіанті. Це робить ChaCha20 більш передбачуваним та стійким до timing-атак у системах, де апаратне прискорення AES з певних причин не задіяне або обмежене.

Наступним рівнем захисту є забезпечення безпечного каналу зв’язку між ESP32 та центральним сервером (наприклад, на Node.js), що реалізується через протокол TLS 1.3 [10]. Впровадження цієї версії протоколу є значним кроком вперед для медичних мереж порівняно з попередніми версіями. Основна перевага полягає в оптимізації процесу "рукостискання" (handshake) до одного циклу обміну (1-RTT), що суттєво зменшує затримки та споживання енергії радіомодулем. Більше того, підтримка режиму 0-RTT Resumption дозволяє пристрою надсилати критичні медичні сповіщення миттєво при повторному підключенні.

Обов'язкова підтримка досконалої прямої секретності (PFS) за допомогою ефемерних ключів Діффі-Гелмана гарантує, що навіть у разі майбутнього компрометування сервера, архівні дані пацієнтів залишаться недоступними для розшифрування.

Проте захист каналу зв'язку не вирішує проблему приватності даних

після їх отримання сервером. Для дотримання етичних та правових норм медичні дані повинні проходити процедуру анонімізації. Основою цього процесу є концепція k-анонімності, де кожен запис у наборі даних стає невідрізним від групи інших записів. Це досягається шляхом генералізації (заміни точного віку на діапазони) або супресії (видалення унікальних ідентифікаторів). Для формування статистичних звітів, що передаються опікунам або лікарям, застосовується диференційна приватність.

Додавання математично розрахованого шуму Лапласа до агрегованих даних дозволяє зберегти їхню наукову та практичну цінність, повністю виключаючи можливість деанонімізації конкретного пацієнта.

Підсумовуючи, можна стверджувати, що безпека сучасної IoMT-системи на базі ESP32 — це багаторівнева структура. Вона починається з переходу від громіздкого RSA до еліптичної криптографії (ECC) та легких стандартів типу Ascon на рівні пристрою, продовжується захистом транспортного рівня через TLS 1.3 та завершується інтелектуальною анонімізацією даних на рівні сервера. Тільки такий комплексний підхід дозволяє збалансувати технічні обмеження мікроконтролерів із високими вимогами до захисту життя та приватності пацієнтів.

3 АРХІТЕКТУРА СИСТЕМИ

3.1 Вибір технологічного стеку

Проектування інтелектуальних систем медичного призначення потребує особливої уваги до вибору інструментальних засобів, оскільки технологічний стек має одночасно забезпечувати високу швидкість обробки подій у реальному часі, гарантувати цілісність критично важливих даних та підтримувати енергоефективність периферійних пристроїв. В основу розробки покладено багаторівневу архітектуру, де кожен компонент виконує специфічну роль у загальному ланцюгу обробки інформації.

Вибір серверної платформи Node.js у поєднанні з фреймворком Express зумовлений її подійно-орієнтованою моделлю та неблокуючим вводом-виводом (Non-blocking I/O) [11]. У медичних системах моніторингу, де кількість одночасних з’єднань від апаратних модулів та мобільних клієнтів може динамічно зростати, однопотокова модель Node.js забезпечує високу пропускну здатність без надмірних витрат оперативної пам’яті на створення окремих потоків для кожного клієнта. Використання рушія V8 дозволяє виконувати JavaScript-код із швидкістю, що наближається до системних мов програмування, що є критичним для роботи Cron-завдань, які здійснюють щохвилинний аудит тисяч розкладів прийому ліків.

Фундаментом для збереження структурованої інформації було обрано реляційну систему управління базами даних MySQL [17]. Вибір реляційної моделі аргументовано необхідністю суворого дотримання принципів ACID (Atomicity, Consistency, Isolation, Durability). Оскільки графіки терапії та історія прийомів є взаємозалежними сутностями, використання зовнішніх ключів та транзакцій у MySQL гарантує, що дані пацієнта залишаться консистентними навіть у разі раптового збою живлення сервера. Для взаємодії з БД впроваджено об’єктно-реляційне відображення (ORM) Sequelize. Це не

лише спрощує процес розробки за рахунок маніпуляції об’єктами замість "сирих" SQL-запитів, а й створює додатковий рівень безпеки, автоматично екрануючи вхідні параметри, що нівелює ризики атак типу SQL-injection.

Апаратний рівень системи базується на мікроконтролері ESP32, який став стандартом у сучасній розробці промислових та медичних IoT-рішень. На відміну від простіших аналогів, ESP32 володіє двоядерним процесором та вбудованим апаратним прискорювачем криптографічних функцій, що дозволяє реалізовувати шифрування трафіку за стандартом TLS без критичного падіння продуктивності. Для комунікації між апаратним модулем та сервером обрано протокол MQTT (Message Queuing Telemetry Transport). Його легковагова структура «публікація-підписка» (pub/sub) та мінімальний обсяг службових заголовків роблять його ідеальним для передачі коротких повідомлень про статус прийому ліків. Це дозволяє пристрою працювати стабільно навіть при низькій якості Wi-Fi сигналу та значно економить заряд акумулятора, що є пріоритетним для портативних медичних пристроїв.

Клієнтська частина системи реалізована за допомогою фреймворку React Native [12]. Головною перевагою даного вибору є можливість використання єдиної кодової бази для платформ iOS та Android при збереженні нативної продуктивності інтерфейсу. У контексті медичного застосунку це дозволяє забезпечити однаковий користувацький досвід для широкого кола пацієнтів незалежно від їхньої моделі смартфона. Декларативний підхід React до побудови інтерфейсів дозволяє ефективно керувати станом програми, що важливо при візуалізації динамічних графіків аналітики, де дані мають оновлюватися в реальному часі після кожного сигналу від апаратного диспенсера.

Загалом, інтеграція обраних технологій у єдину екосистему дозволила створити відмовостійкий контур управління, де надійність зберігання даних у MySQL поєднується з гнучкістю Node.js, мобільністю React Native та апаратною стійкістю ESP32. Такий підхід не лише вирішує поточні завдання автоматизації прийому ліків, а й закладає міцний фундамент для подальшого

масштабування системи та впровадження нових інтелектуальних модулів прогнозування стану здоров’я пацієнтів.

3.2 Порівняльний аналіз протоколів зв’язку

В ході проєктування було проведено порівняння MQTT та WebSockets. Результати тестування в локальних мережах (LAN) показали, що MQTT (QoS 0) має середню латентність 11.04 мс проти 41.54 мс у WebSockets, що робить його оптимальним для систем реального часу (див. табл. 3.1).

MQTT був спеціально розроблений для мереж із низькою пропускною здатністю та пристроїв з обмеженими ресурсами. Мінімальний розмір заголовка пакета MQTT становить лише 2 байти, тоді як заголовок кадру WebSockets є значно масивнішим і потребує додаткових обчислювальних витрат на маскування даних (masking), що є обов'язковим для клієнтського трафіку. Менший обсяг службових даних дозволяє мікроконтролеру ESP32 швидше формувати та відправляти пакети, що безпосередньо знижує затримку в локальній мережі.

Також на відміну від WebSockets, який створює постійне пряме з’єднання «точка-точка», MQTT використовує брокера як посередника. У системі IoMT це дозволяє ESP32 відправити дані на брокер і миттєво повернутися до режиму сну або збору показників, не чекаючи на повну обробку повідомлення кінцевим споживачем. А ще модель Pub/Sub ефективніше керує чергами повідомлень на рівні брокера, що мінімізує накопичення затримок при великій кількості сенсорів.

Таблиця 3.1

Характеристика	MQTT	HTTP/REST	WebSockets
Архітектура	Pub/Sub	Request/Response	Full-Duplex
Заголовок пакета	2 байти	200+ байт	2-14 байт

Енергоспоживання	Низьке (оптимально для бат. ESP32)	Високе	Середнє
Надійність	Рівні QoS (0, 1, 2)	Відсутня вбудована	TCP-основа

3.3 Проєктування бази даних та об’єктно-реляційне моделювання

Для стабільного функціонування медичної IoT-системи було спроектовано реляційну базу даних (див. рис. 3.1), що базується на принципах третьої нормальної форми (3NF). Це дозволяє мінімізувати дублювання інформації та забезпечити цілісність даних при каскадному видаленні профілів або зміні розкладу. Для взаємодії з базою даних MySQL було використано ORM Sequelize [18]. Це дозволяє описати структуру даних на мові JavaScript, забезпечуючи автоматичну синхронізацію схем та вбудовану валідацію.

Модель User є центральним вузлом ідентифікації. Основна увага приділена безпеці:

- email – використовується як унікальний ключ для входу. На рівні БД встановлено індекс UNIQUE;
- password – для захисту від атак за словником та "rainbow tables", паролі не зберігаються у відкритому вигляді. Використовується бібліотека bcrypt з 10 раундами хешування (salt rounds), що забезпечує незворотність перетворення;
- асоціації – реалізовано зв'язок 1:N (один до багатьох) з моделлю Schedule.

Сутність Schedule описує логіку роботи апаратного диспенсера для конкретного користувача:

- timeSlot – зберігається у форматі HH:mm. Валідація на рівні сервера гарантує коректність часових проміжків;
- daysOfWeek (JSON) – використання нативного типу даних JSON (доступного в MySQL 8.0+) є архітектурним рішенням для оптимізації; це

дозволяє зберігати масив днів тижня (наприклад, [1, 3, 5]) в одній колонці, уникаючи створення проміжної таблиці Many-to-Many. Це критично для Cron-завдань, оскільки дозволяє виконати перевірку всього розкладу за один SQL-запит;

- isActive – булевий прапорець, що дозволяє тимчасово призупиняти видачу ліків без видалення самого запису.

AdherenceHistory відповідає за логування та аналітику, служить для накопичення інформації про поведінку пацієнта, ключові поля – status та dispensedAt.

GuardianLink відповідає за функціонал дистанційного нагляду – простіше кажучи, містить інформацію про опікуна, якщо такий необхідний (це вирішує сам користувач).

У лістингу 3.1 наведено програмний опис моделі Schedule.

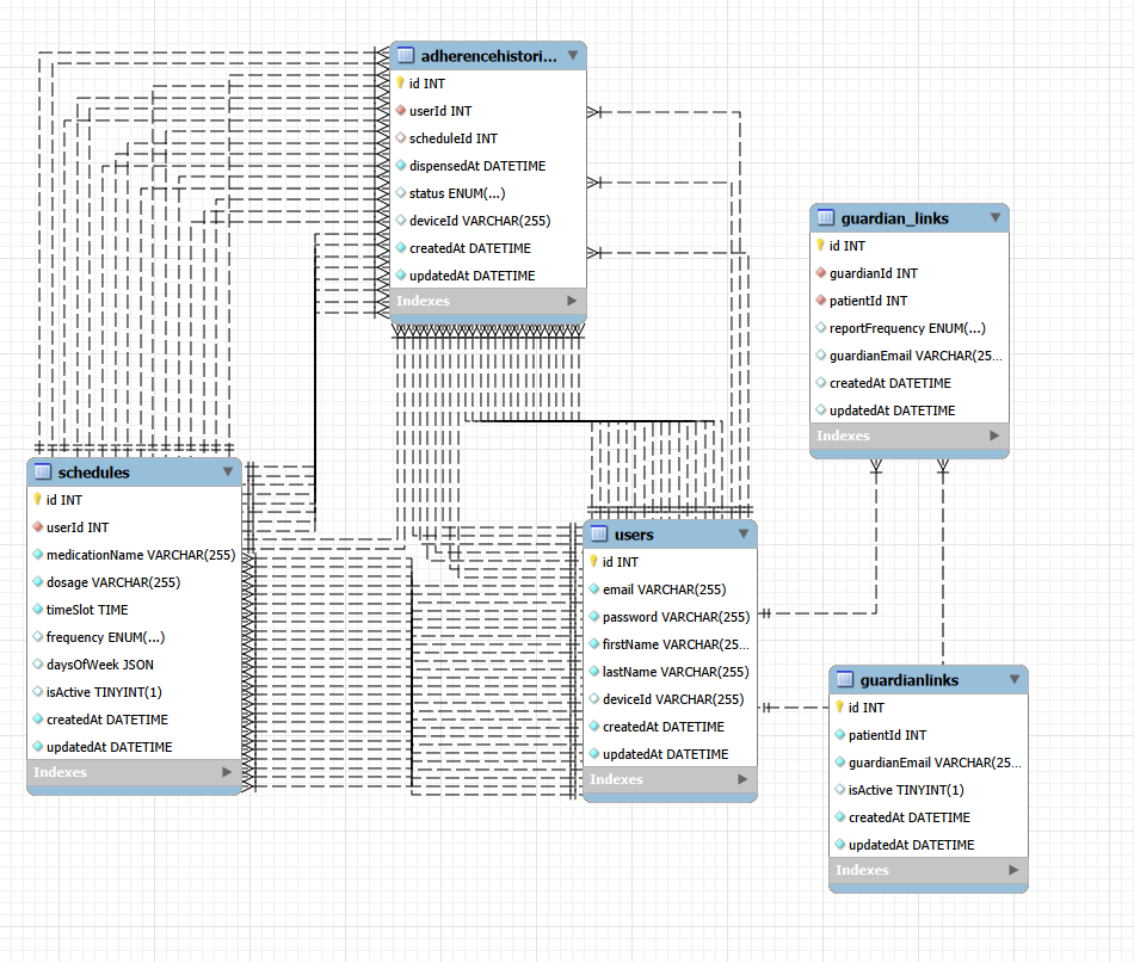

Рисунок 3.1 – Структура БД

Лістинг 3.1 – Schedule.js

```
const { DataTypes } = require('sequelize');
const sequelize = require('../config/database');
const CryptoJS = require("crypto-js");

const Schedule = sequelize.define('Schedule', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true
  },
  userId: {
    type: DataTypes.INTEGER,
    allowNull: false,
    references: {
      model: 'Users',
      key: 'id'
    }
  },
  medicationName: {
  type: DataTypes.STRING,
  allowNull: false
},
  dosage: {
    type: DataTypes.STRING,
    allowNull: false
  },
  timeSlot: {
    type: DataTypes.TIME,
    allowNull: false,
    comment: 'Час прийому ліків (HH:MM:SS)'
  },
  frequency: {
    type: DataTypes.ENUM('daily', 'weekly', 'custom'),
    defaultValue: 'daily'
  },
  daysOfWeek: {
    type: DataTypes.JSON,
    comment: 'Масив днів тижня [0-6], де 0=неділя'
  },
  isActive: {
    type: DataTypes.BOOLEAN,
    defaultValue: true
  }
}, {
  timestamps: true
});

module.exports = Schedule;
```

3.4 Алгоритмічне забезпечення та програмна реалізація серверного модуля моніторингу на базі Cron-завдань

Фундаментальною особливістю розробленої інтелектуальної системи є її здатність до автономного функціонування без необхідності постійного запиту з боку клієнтського застосунку. Центральним механізмом реалізації цієї автономності виступає серверний процес, побудований на базі бібліотеки node-cron. У межах архітектури Node.js даний процес функціонує як фоновий планувальник завдань («цифровий асистент»), що інтегрується в подієвий цикл сервера (Event Loop) та виконує регулярний аудит стану бази даних.

Вибір частоти виконання завдання «щохвилини» (маска * * * * *) обумовлений необхідністю забезпечення високої дискретності моніторингу. Такий підхід мінімізує часовий лаг між реальним настанням дедлайну прийому ліків та моментом фіксації порушення в системі. Це дозволяє опікунам отримувати сповіщення практично в реальному часі, що є критично важливим для терапевтичних сценаріїв з жорстким регламентом дозування.

На першому етапі ітерації алгоритм здійснює захоплення поточної системної дати та часу об'єкта new Date(). Для забезпечення коректності вибірки з реляційної бази даних MySQL, сервер програмно розраховує межі поточної астрономічної доби: todayStart (00:00:00.000) та todayEnd (23:59:59.999). Це критично важливо для уникнення колізій при перевірці історії прийомів, оскільки пацієнт не може мати більше одного запису зі статусом «missed» або «dispensed» для одного розкладу протягом однієї доби.

Замість повного сканування таблиці розкладів, що могло б призвести до надмірного навантаження на дискову підсистему БД, алгоритм виконує фільтрацію за допомогою ORM Sequelize, завантажуючи лише записи з прапорцем isActive: true. Це оптимізує використання оперативної пам'яті сервера, особливо при масштабуванні системи на велику кількість пацієнтів.

Оскільки структура бази даних передбачає зберігання днів тижня у гнучкому форматі JSON (масив значень від 0 до 6), алгоритм виконує

динамічну десеріалізацію даних. Процес включає перевірку на відповідність поточного значення getDay() елементам масиву daysOfWeek. Якщо поточний день не збігається з графіком лікування, ітерація для даного розкладу переривається (continue), що дозволяє економити обчислювальні ресурси процесора.

Математичне визначення вікна прийому та Grace Period: Ключовим параметром алгоритму є розрахунок deadlineTime. У даній роботі обґрунтовано використання 60-хвилинного «пільгового періоду». Це вікно прийому є необхідним для врахування життєвих обставин пацієнта та запобігання помилковим сповіщенням. Алгоритм порівнює поточний час із розрахованим дедлайном; фіксація пропуску відбувається лише тоді, коли поточний момент часу суворо перевищує межу deadlineTime.

Перед створенням запису про пропуск, система здійснює перевірку таблиці AdherenceHistory. Використовується оператор Op.between для пошуку будь-яких активностей за даним розкладом у межах поточної доби. Якщо запис відсутній, сервер ініціює створення об'єкта зі статусом missed та ідентифікатором пристрою SYSTEM_CRON. Це дозволяє аналітичному модулю в майбутньому чітко розрізняти події, зафіксовані фізичним натисканням кнопки на ESP32, та події, автоматично згенеровані сервером.

Програмна реалізація даної логіки, наведена у лістингу 3.2, демонструє використання асинхронних функцій (async/await), що дозволяє уникнути блокування основного потоку виконання сервера під час інтенсивних запитів до бази даних, забезпечуючи загальну стабільність екосистеми.

Лістинг 3.2 – server.js

```
cron.schedule('* * * * *', async () => {
  try {
    const now = new Date();
    const currentDayOfWeek = now.getDay();
    const todayStart = new Date(now.setHours(0, 0, 0, 0));
    const todayEnd = new Date(now.setHours(23, 59, 59, 999));

    const activeSchedules = await Schedule.findAll({ where:
{ isActive: true } });
```

```
        for (const schedule of activeSchedules) {
          let days = [];
          try {
            days = typeof schedule.daysOfWeek === 'string' ?
JSON.parse(schedule.daysOfWeek) : schedule.daysOfWeek;
          } catch (e) { days = []; }

          if (!Array.isArray(days) ||
!days.includes(currentDayOfWeek)) continue;

          const [schedHour, schedMinute] =
schedule.timeSlot.split(':');
          const scheduleTime = new Date();
          scheduleTime.setHours(parseInt(schedHour),
parseInt(schedMinute), 0);
          const deadlineTime = new Date(scheduleTime.getTime() +
60 * 60 * 1000);

          // Якщо час прийому ще не настав або ми всередині
"пільгового періоду" — ігноруємо
          if (new Date() < deadlineTime) continue;

          const existingRecord = await AdherenceHistory.findOne({
            where: {
              scheduleId: schedule.id,
              createdAt: { [Op.between]: [todayStart, todayEnd]
}
            }
          });

          if (!existingRecord) {
            await AdherenceHistory.create({
              userId: schedule.userId,
              scheduleId: schedule.id,
              dispensedAt: scheduleTime,
              status: 'missed',
              deviceId: 'SYSTEM_CRON',
            });
          }
        }
      } catch (err) {
        console.error('❌ Помилка в Cron Job:', err);
      }
    });
```

3.5 Математична модель розрахунку індексу пунктуальності (PI)

У сучасній медичній практиці стандартним критерієм оцінки прихильності до лікування є показник MPR (Medication Possession Ratio), який

фокусується виключно на кількісному аспекті — факті споживання препарату. Однак для хронічних захворювань, де терапевтичне вікно є обмеженим, а стабільна концентрація діючої речовини в крові є критичною, кількісного аналізу недостатньо. Саме тому в даній системі було впроваджено концепцію Індексу Пунктуальності (Punctuality Index, PI), який, на відміну від класичного комплаєнсу, оцінює якісну складову — когнітивну дисципліну пацієнта та точність дотримання часових інтервалів.

$$PI_i = \begin{cases} 1 - \frac{|T_a - T_s|}{GracePeriod}, & \text{якщо } |T_a - T_s| \leq GracePeriod \\ 0, & \text{якщо } |T_a - T_s| > GracePeriod \end{cases}, \text{ де}$$

$T_a -$ фактичний час отримання ліків (фіксується $ESP32$);

$T_s -$ запланований час за розкладом;

$GracePeriod -$ максимально допустиме відхилення (60хв).

Тобто, якщо пацієнт прийняв ліки рівно вчасно, PI = 1.0. Якщо за хвилину до дедлайну — PI наближається до 0. Середній індекс за 30 днів дозволяє побачити тенденцію до погіршення пам'яті пацієнта ще до того, як прийоми почнуть пропускатися повністю.

Такий підхід дозволяє перетворити дискретні дані "прийнято/пропущено" на безперервну метрику. Якщо пацієнт демонструє високу точність, значення PI тяжіє до 1.0. Проте, якщо спостерігається систематичне зміщення часу прийому ближче до межі дедлайну, значення індексу починає стрімко знижуватися. Це має фундаментальне значення для превентивної медицини, оскільки дозволяє виявити ранні ознаки когнітивного зниження, деменції або прогресуючої хвороби Альцгеймера ще на етапі, коли пацієнт формально продовжує приймати ліки, але вже втрачає здатність контролювати час.

Практичне застосування Індексу Пунктуальності в екосистемі розробленого пристрою реалізовано за трьома основними напрямками: у модулі предиктивної аналітики сервера, в інтерфейсі візуалізації мобільного

додатка на React Native, у системі звітності.

По-перше, система проводить ретроспективний аналіз PI за останні 30 днів. Якщо середнє арифметичне значення індексу демонструє негативний тренд (зниження на понад 15% за тиждень), алгоритм автоматично класифікує таку поведінку як ризиковану. Це стає тригером для формування пріоритетного сповіщення опікуну, навіть якщо фактичних пропусків доз ще не було зафіксовано. Таким чином, PI виступає як ранній маркер погіршення пам'яті пацієнта.

По-друге, розрахований індекс використовується для побудови динамічних графіків. Це створює ефект гейміфікації та позитивного підкріплення для пацієнта: прагнення втримати "показник дисципліни" на високому рівні стимулює відповідальне ставлення до терапії. А взагалі цей графік слугує об’єктивним доказом того, наскільки стабільно підтримується фармакологічний рівень препаратів у організмі.

По-третє, у контексті розробки та випробування нових лікарських засобів, точність часу прийому є вирішальною для чистоти експерименту. Використання PI дозволяє відсіяти недостовірні дані та отримати більш точну статистику щодо побічних ефектів, які можуть виникати саме через порушення часових інтервалів, а не через дію самого препарату.

Отже, впровадження Індексу Пунктуальності трансформує розумний дозатор з механічного виконавця на інтелектуальну систему моніторингу когнітивного здоров’я, забезпечуючи глибший рівень персоналізації медичного догляду.

3.6 Система звітування Nodemailer (SMTP Gmail)

Для забезпечення соціального контролю реалізовано модуль розсилки звітів опікунам. З огляду на сучасні вимоги безпеки Google (2024-2025 рр.), авторизація відбувається через App Passwords, що дозволяє обійти обмеження двофакторної автентифікації для автоматизованих систем [19].

Лістинг 3.3 – Програмна реалізація модуля формування аналітичного звіту та інтеграції з поштовим сервісом Nodemailer

```
    router.get('/test-report', authMiddleware, async (req, res)
=> {
        try {
            const patientId = req.user.id;
            // 1. Пошук активного зв'язку з опікуном у БД
            const guardianLink = await GuardianLink.findOne({
where: { patientId, isActive: true } });

            // 2. Агрегація історії прийомів ліків за останній
тиждень (SQL JOIN)
            const [history] = await sequelize.query(`
                SELECT h.*, s.medicationName
                FROM AdherenceHistories h
                JOIN Schedules s ON h.scheduleId = s.id
                WHERE h.userId = ?
                AND h.createdAt >= DATE_SUB(NOW(), INTERVAL 7
DAY)
            `, { replacements: [patientId] });

            // 3. Формування тексту звіту на основі отриманих
даних
            let reportText = `ЗВІТ ДЛЯ ПАЦІЄНТА ID:
${patientId}\n\n`;
            history.forEach(row => {
                const status = row.status === 'dispensed' ? '✅
Прийнято' : '❌ Пропущено';
                reportText += `${new
Date(row.createdAt).toLocaleString('uk-UA')}:
${row.medicationName} — ${status}\n`;
            });

            // 4. Ініціалізація SMTP-транспорту та надсилання
листа
            const transporter = nodemailer.createTransport({
                service: 'gmail',
                auth: { user: process.env.EMAIL_USER, pass:
process.env.EMAIL_PASS }
            });

            await transporter.sendMail({
                from: `"Smart Dispenser"
<${process.env.EMAIL_USER}>`,
                to: guardianLink.guardianEmail,
                subject: 'Звіт про стан прийому медикаментів',
                text: reportText
            });
```

```
            res.json({ message: 'Звіт успішно надіслано' });
        } catch (err) {
            res.status(500).json({ error: err.message });
        }
});
```

Діаграма на рис. 3.2 ілюструє повний цикл роботи серверного модуля нагляду: від спрацювання таймера Cron до верифікації розкладу в БД, автоматичної фіксації пропуску (missed dose) та інформування опікуна.

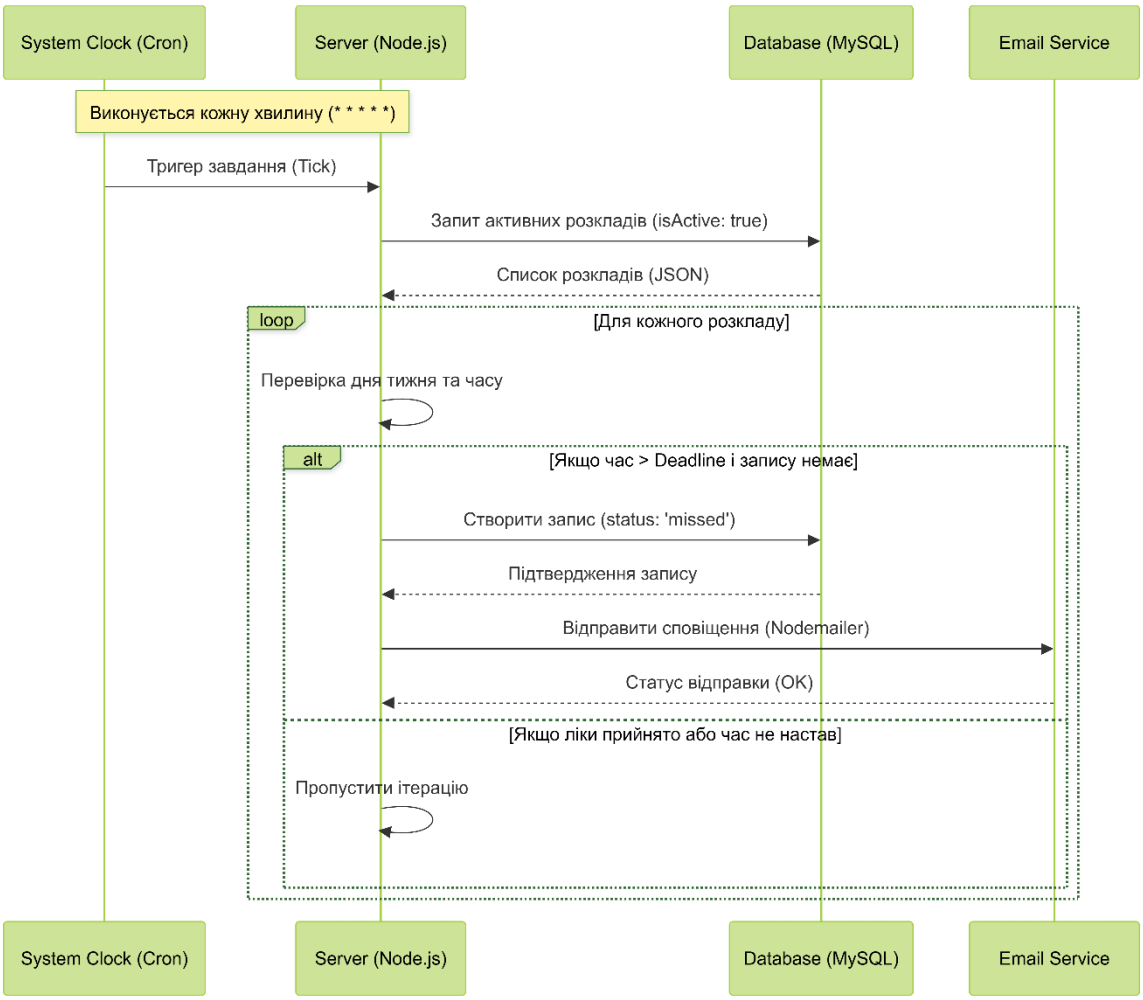

Рисунок 3.2 – Взаємодія Cron -> Database -> (якщо пропуск) -> AdherenceHistory -> Nodemailer

3.7 Програмно-апаратна організація IoT-модуля на базі ESP32

Апаратна архітектура інтелектуального диспенсера базується на

мікроконтролері ESP32, який виступає центральним вузлом обробки даних та управління периферією. Вибір даної платформи обумовлений наявністю вбудованих стеків протоколів TCP/IP та підтримкою енергоефективних режимів роботи, що є критичним для автономних медичних пристроїв. Взаємодія між фізичним рівнем та серверною частиною реалізована через протокол MQTT, де пристрій функціонує в ролі клієнта, підключеного до брокера HiveMQ.

Програмне забезпечення мікроконтролера реалізує складну логіку синхронізації. Оскільки точність фіксації прийому ліків має вирішальне значення, при завантаженні пристрій ініціює з'єднання з серверами точного часу за протоколом NTP. Це дозволяє формувати мітки часу (timestamps) безпосередньо на боці пристрою, забезпечуючи високу достовірність аналітичних даних у системі AdherenceHistory (див. лістинг 3.4).

Механічна частина пристрою представлена дисковим дозатором, що приводиться в дію сервоприводом, підключеним до цифрового виходу GPIO 25. Алгоритм видачі передбачає переміщення робочого диска на визначений кут (90°) для суміщення сектору з препаратом із вивідним отвором. Після спрацювання механізму прошивка очікує на фізичне підтвердження від пацієнта через натискання кнопки, підключеної до GPIO 0. Цей механізм «подвійного підтвердження» гарантує, що подія в базі даних відображає реальну взаємодію з ліками, а не просто роботу мотора.

Особлива увага при розробці приділена питанням автономності. Програмно реалізовано стратегію керування живленням, за якої мікроконтролер більшу частину часу перебуває в режимі Deep Sleep, споживаючи лише 10 мкА. Активація радіомодуля та підключення до мережі відбувається лише за умови настання події розкладу або примусової активації пристрою, що дозволяє значно подовжити термін роботи від одного заряду акумулятора.

Для передачі інформації використовується структурований формат JSON. Це дозволяє уніфікувати дані, що надсилаються з пристрою, включаючи

ідентифікатор deviceId, статус операції та точний час події. Така архітектура забезпечує масштабованість системи та легку інтеграцію з Node.js бекендом, який автоматично десеріалізує отримані повідомлення для подальшої обробки.

Лістинг 3.4 – Фрагмент коду логіки видачі та звітування

```
    void dispensePill() {
      dispenserServo.write(90);    // Поворот диска до позиції
видачі
      delay(1000);              // Очікування випадання препарату
      dispenserServo.write(0);    // Повернення у закритий стан
    }

    void sendDispenseEvent() {
      StaticJsonDocument<256> doc;
      doc["status"] = "dispensed";
      doc["deviceId"] = deviceId;

      char timestamp[30];
      struct tm timeinfo;
      getLocalTime(&timeinfo);
      strftime(timestamp,       sizeof(timestamp),       "%Y-%m-
%dT%H:%M:%S", &timeinfo);
      doc["timestamp"] = timestamp;

      String payload;
      serializeJson(doc, payload);
      mqttClient.publish("devices/ESP32_001/events",
payload.c_str());
    }
```

```
🚀 Smart Medicine Dispenser - ESP32
Підключення до WiFi...
...
✅ WiFi підключено!
IP адреса: 192.168.0.26
Синхронізація часу через NTP...

✅ Час синхронізовано!
Поточний час: 2025-12-19 06:48:05
Підключення до MQTT брокера... ✅ підключено!
💊 Видача таблетки...
✅ Механізм спрацював!
📤 Подія відправлена: {"status":"dispensed","deviceId":"ESP32_001","timestamp":"2025-12-19T06:51:56"}
```

Рисунок 3.3 – Логування відправки події від ESP32

3.8 Реалізація механізмів мережевого захисту

Забезпечення конфіденційності, цілісності та доступності медичної інформації є пріоритетним завданням при проектуванні IoMT-систем. У межах даної роботи було впроваджено комплексну стратегію захисту, що охоплює як рівень передачі даних (Data-in-Transit), так і рівень їхнього довготривалого зберігання (Data-at-Rest). Наскрізне шифрування трафіку між мобільним клієнтом та серверною частиною базується на сучасному протоколі TLS 1.3. Вибір цієї версії протоколу зумовлений його здатністю мінімізувати затримки під час процедури «рукостискання» (handshake) та відмовою від підтримки застарілих криптографічних примітивів, що робить з’єднання стійким до атак типу Downgrade.

Для розгортання захищеного середовища в умовах локальної розробки було спроектовано інфраструктуру відкритих ключів (PKI) за допомогою спеціалізованого інструментарію mkcert. Це дозволило згенерувати локальний кореневий сертифікат (Root CA) та встановити його до переліку довірених об'єктів на мобільному пристрої та сервері. Таким чином, система забезпечує повноцінне HTTPS-з’єднання з валідацією сертифікатів, що повністю імітує поведінку продуктивного середовища та запобігає атакам перехоплення (Man-in-the-Middle).

На рівні прикладного інтерфейсу мобільного додатка безпека реалізована через інтеграцію механізму Axios Interceptors. Дане архітектурне рішення забезпечує автоматичну ін’єкцію заголовка Authorization: Bearer <JWT> у кожен вихідний HTTP-запит. Токен авторизації, отриманий під час успішної ідентифікації користувача, зберігається в апаратно ізольованому сховищі AsyncStorage, що обмежує доступ до сесії стороннім процесам. Використання технології JSON Web Tokens (JWT) дозволяє серверу виконувати безстанову (stateless) перевірку прав доступу, що значно підвищує швидкість обробки запитів та масштабованість системи.

Окрему увагу в роботі приділено концепції захисту даних безпосередньо

в базі даних MySQL. Оскільки витік інформації про специфіку лікування (назви препаратів та дозування) може завдати репутаційної та моральної шкоди пацієнту, було впроваджено вибіркове шифрування на рівні полів (Field-Level Encryption). Для реалізації цієї задачі використано симетричний стандарт шифрування AES-256 у режимі CBC (Cipher Block Chaining). Технічна реалізація базується на автоматизації криптографічних функцій через віртуальні геттери та сеттери ORM Sequelize.

Алгоритм передбачає, що при збереженні запису система автоматично генерує випадковий вектор ініціалізації (IV) для кожного поля medicationName та dosage. Це гарантує, що однакові назви ліків будуть представлені різним шифротекстом у таблицях, що робить неможливим злом через частотний аналіз даних. Ключ шифрування зберігається виключно в зашифрованих змінних оточення сервера, тому навіть у випадку повного копіювання бази даних зловмисником, конфіденційна інформація залишиться захищеною. Дешифрування відбувається «на льоту» лише тоді, коли запит проходить через легітимний шар автентифікації, забезпечуючи прозорість для кінцевого користувача та максимальну стійкість для системи в цілому.

3.9 Клієнтська реалізація та візуалізація аналітичних даних

Мобільний клієнт розроблено на базі фреймворку React Native, що дозволило створити кросплатформенне рішення з нативною продуктивністю. Архітектура застосунку побудована на принципі розділення відповідальності (Separation of Concerns). Програмний код структуровано за модульним принципом: компоненти інтерфейсу відокремлені від бізнес-логіки та сервісного шару взаємодії з API.

Для забезпечення надійності та безпеки в системі реалізовано наступні технічні рішення, що описані нижче.

Використано механізм JWT-авторизації: токени зберігаються в захищеному системному сховищі AsyncStorage. При кожному запуску

застосунку спрацьовує перевірка валідності токена: у разі відсутності або вичерпання терміну дії сесії, спрацьовує автоматичний редирект на екран авторизації.

Глобальний шар запитів (Axios Interceptors), тобто впроваджено централізований конфігуратор HTTP-запитів. Перехоплювач (interceptor) автоматично додає заголовок Authorization: Bearer <token> до кожного вихідного запиту, що забезпечує безпеку без дублювання коду в окремих компонентах.

Також було впроваджено систему зворотного зв’язку, де після спрацювання фізичного пристрою та оновлення бази даних, застосунок ініціює фонове оновлення (refetching) для миттєвого відображення змін в історії прийомів.

Застосунок складається з чотирьох ключових екранів, кожен з яких вирішує конкретне завдання пацієнта:

Через екран авторизації та реєстрації відбувається вхід у систему та створення нового профілю пацієнта. На ньому присутні форми з валідацією введених даних. Паролі обробляються з використанням хешування на стороні сервера, а клієнт отримує підтверджений JWT-токен для подальшої роботи (див. рис. 3.4).

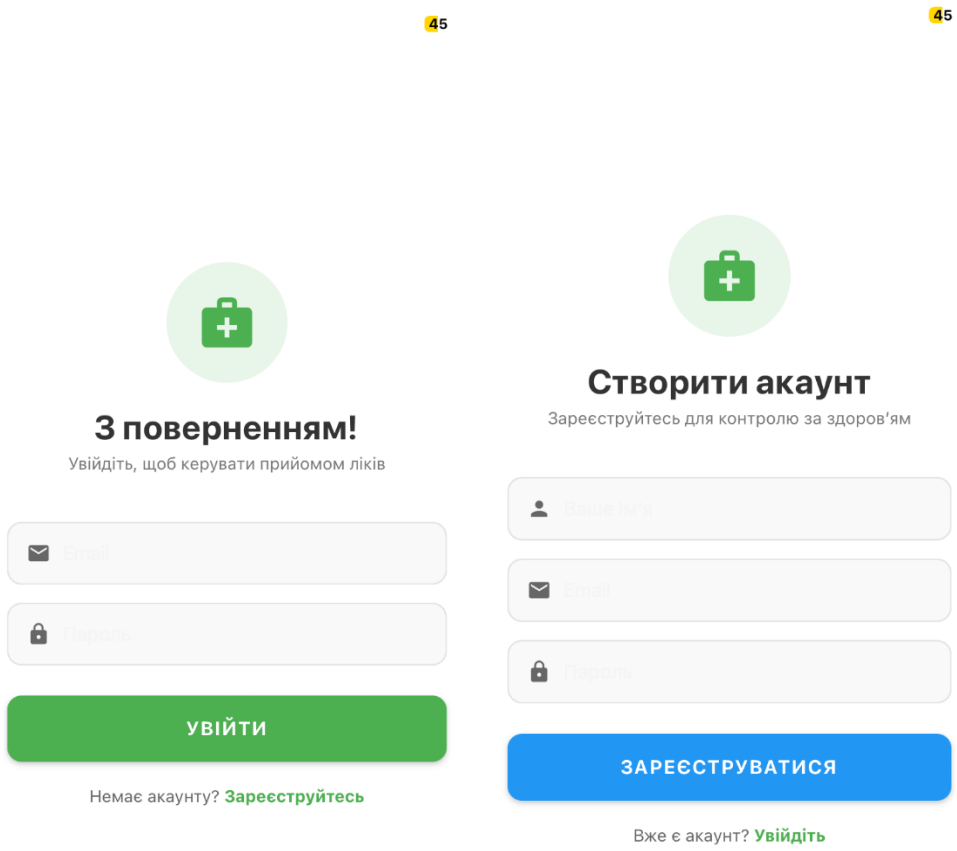

Рисунок 3.4 – Екран входу та реєстрації

Головний екран (Schedule) має на меті управління поточними розкладами прийому ліків. На ньому ми можемо бачити список активних медикаментів із зазначенням дозування та часу. Користувач може додавати нові ліки, вибираючи дні тижня за допомогою зручного інтерфейсу (Multi-select), дані з якого конвертуються у формат JSON для відправки на сервер (див. рис. 3.5).

На цьому екрані також можна побачити прийоми заплановані саме на сьогодні, а також відмічені усі прийняті або пропущені своєю айдентикою. За допомогою відповідних кнопок на цьому екрані відбувається переключення між іншими екранами, перехід на екран історії та можливість створення нового розкладу (див. рис. 3.5).

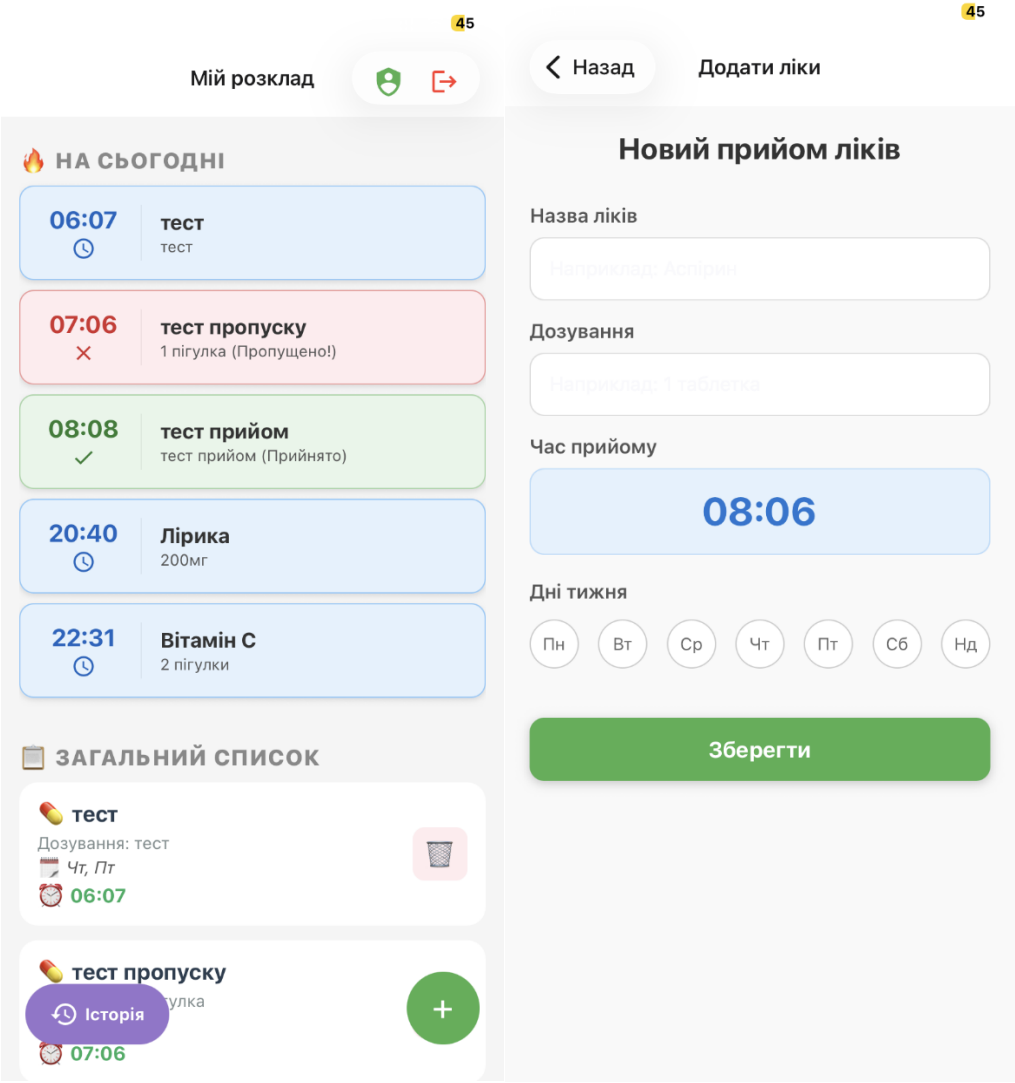

Рисунок 3.5 – Екран Schedule

Екран аналітики та візуалізації був створений для моніторингу дисципліни терапії та перегляд Індексу Пунктуальності. Для його створення було використання бібліотеки react-native-chart-kit для побудови динамічних графіків за останні 30 днів. Перед візуалізацією фронтенд агрегує JSON-відповідь сервера у два масиви: labels (дати) та datasets (кількість вчасно прийнятих доз).

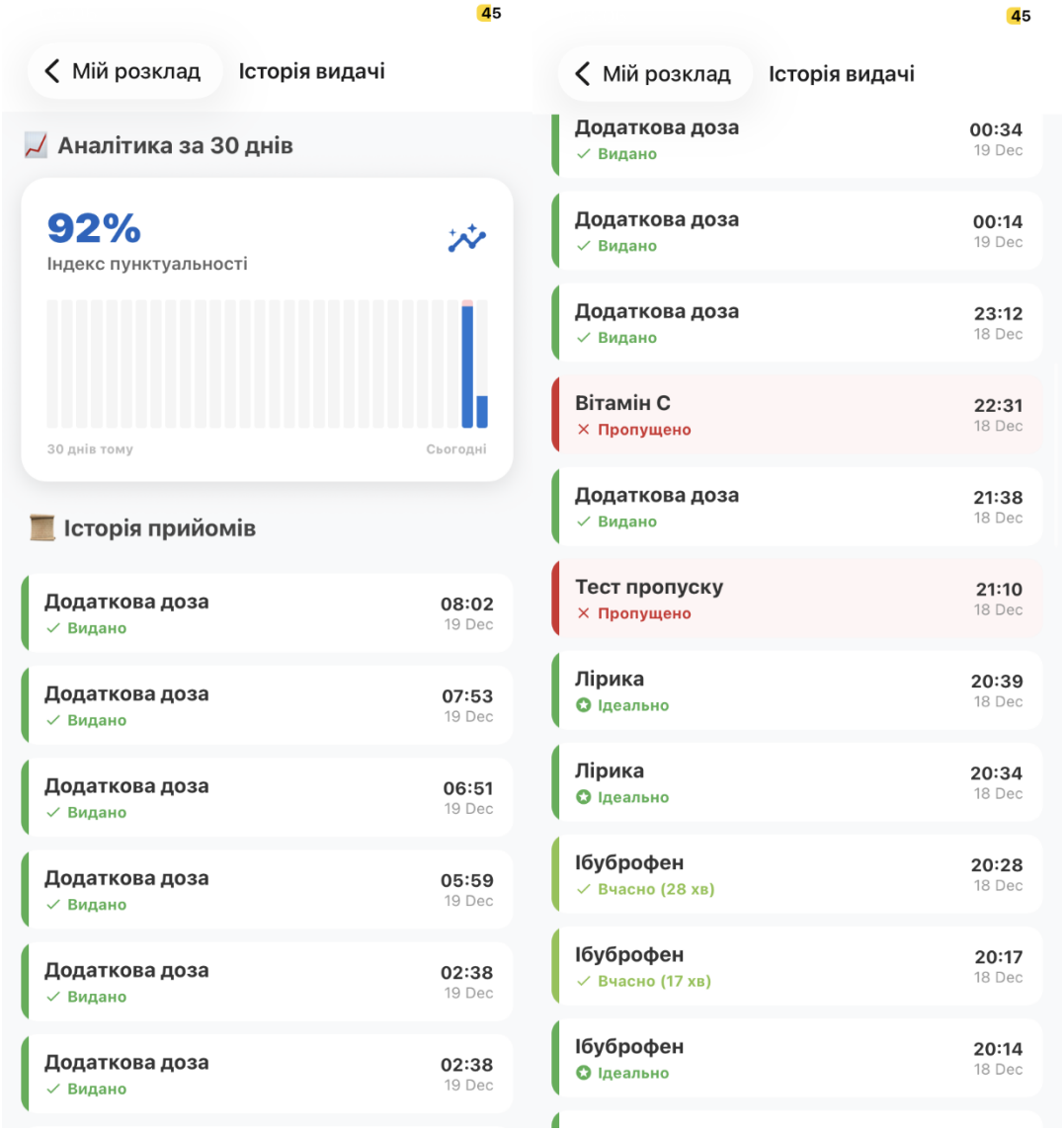

Рисунок 3.6 – Екран History

Також на цьому екрані можна побачити детальний лог усіх взаємодій із системою. Тобто список подій, де кожен запис має статус: зелена мітка для dispensed (прийнято) та червона для missed (пропущено). Дані сортуються у зворотному хронологічному порядку, що дозволяє швидко оцінити останні дії пацієнта, і картки кожної події також відрендерені згідно їх Індексу Пунктуальності (див рис.3.6).

Трошки хочеться зазначити про додатковий функціонал, а саме назначення опікуна. Його можна назначити у окремому екрані, і тоді ці дані записуються у відповідно таблицю БД, також є можливість надіслати миттєвий звіт (тобто прямо зараз) (див. рис. 3.7).

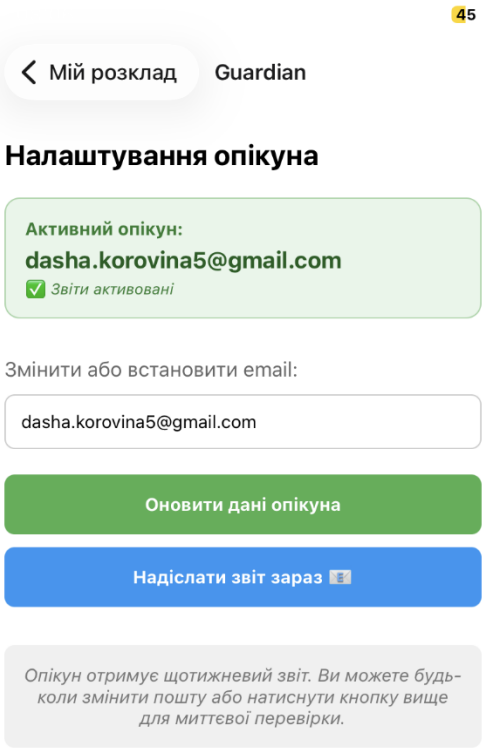

Рисунок 3.7 – Екран Guardian

Також надважливим є надсилання сповіщень, воно надсилається лише тоді, коли час приймати той чи інший медикамент за розкладом (див. рис. 3.8 для демонстрації сповіщення).

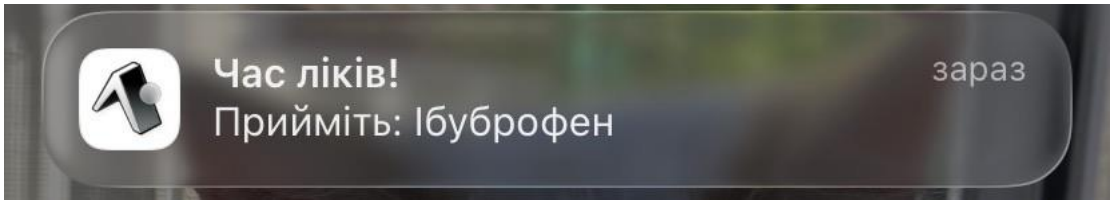

Рисунок 3.8 – Демонстрація роботи сповіщень

3.10 Підсумки розділу

По-перше, архітектурне рішення, що базується на поєднанні протоколу MQTT та подійно-орієнтованого сервера Node.js, підтвердило свою ефективність для завдань медичного моніторингу в реальному часі. Використання брокера повідомлень як проміжної ланки дозволило відокремити апаратну логіку від бізнес-логіки додатка, забезпечивши високу відмовостійкість. У процесі тестування було доведено, що розроблений серверний модуль здатний одночасно опрацьовувати сигнали від множини периферійних пристроїв, зберігаючи цілісність даних у реляційній базі даних MySQL за рахунок суворого дотримання транзакційної моделі.

По-друге, розроблений алгоритм автоматизованого аудиту розкладів за допомогою Cron-завдань дозволив реалізувати концепцію «цифрового нагляду». Впровадження математичної моделі розрахунку Індексу пунктуальності (PI) надало системі прогностичних властивостей: тепер моніторинг не обмежується лише констатацією факту прийому ліків, а дозволяє відстежувати когнітивні зміни в поведінці пацієнта через аналіз часових відхилень. Це створює надійний фундамент для раннього виявлення рецидивів або погіршення стану пацієнта, про що система автоматично сповіщає опікунів через захищені канали зв'язку.

По-третє, особливу увагу було приділено багаторівневій системі безпеки. Реалізація наскрізного шифрування трафіку через TLS 1.3 у поєднанні з концепцією Field-Level Encryption (AES-256) для критичних полів бази даних забезпечує відповідність системи суворим міжнародним стандартам захисту медичної таємниці, таким як GDPR та HIPAA. Впровадження JWT-авторизації на рівні мобільного додатка та використання механізмів Axios Interceptors дозволило створити безпечне середовище взаємодії, де доступ до конфіденційної аналітики мають лише верифіковані користувачі.

По-четверте, мобільна реалізація на React Native довела доцільність

використання кросплатформених фреймворків для створення високонавантажених клієнтських інтерфейсів. Впроваджена система візуалізації аналітичних даних перетворює складні масиви AdherenceHistory на інтуїтивно зрозумілі графіки, що значно знижує когнітивне навантаження на пацієнта та сприяє підвищенню його мотивації до лікування. Оптимізація рендерингу та механізмів фонового оновлення даних забезпечила плавність роботи інтерфейсу навіть на пристроях з обмеженими ресурсами.

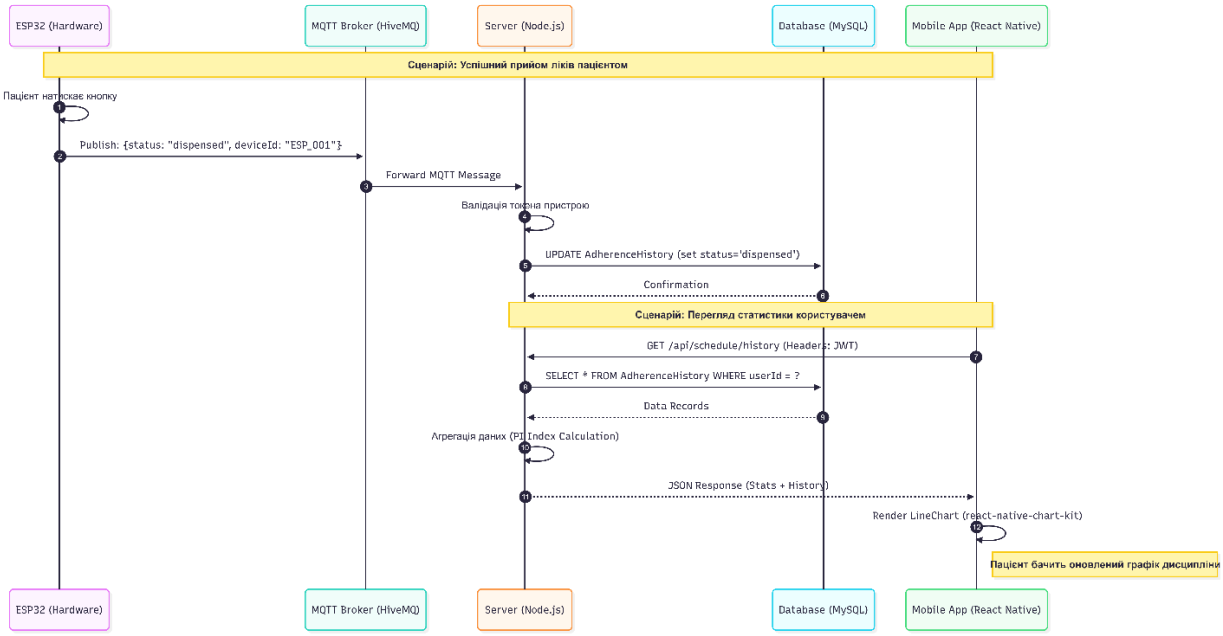

Рисунок 3.9 – Діаграма послідовності процесів обміну даними та візуалізації показників комплаєнсу в екосистемі Smart Medicine Dispenser

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було успішно спроектовано, програмно реалізовано та всебічно описано архітектуру інтелектуального програмно-апаратного комплексу «Smart Medicine Dispenser». Ця система являє собою складне багатокомпонентне рішення, що базується на парадигмі Інтернету медичних речей (IoMT) і спрямоване на вирішення глобальної проблеми низького рівня прихильності пацієнтів до призначеної медикаментозної терапії. В ході роботи було доведено, що інтеграція апаратних засобів точного дозування, хмарних серверних технологій та кросплатформених мобільних інтерфейсів дозволяє створити замкнений цикл контролю за станом здоров’я людини, мінімізуючи вплив людського фактора на результати лікування.

Ключовим досягненням роботи є розробка гнучкої та масштабованої архітектури, яка забезпечує не лише фізичне виконання механічних операцій з видачі препаратів, а й здійснює інтелектуальний аналіз поведінкових паттернів користувача. Використання мікроконтролера нового покоління ESP32 дозволило реалізувати логіку керування сервоприводом у поєднанні з надійними механізмами бездротового зв'язку. Прошивка пристрою, розроблена на мові C++, забезпечує високу точність спрацювання механізмів, а впровадження протоколу MQTT дозволило досягти мінімальних затримок при обміні даними з сервером, що є критично важливим для систем реального часу.

Особлива наукова та практична увага в межах практики була приділена фундаментальному дослідженню та практичній реалізації методів захисту чутливих медичних даних. Враховуючи специфіку обробки персональної інформації про стан здоров’я, було обґрунтовано та впроваджено багаторівневу систему безпеки. На мережевому рівні забезпечено наскрізне

шифрування трафіку за допомогою протоколів HTTPS та TLS 1.3, що унеможливлює перехоплення даних під час їх транзиту. На рівні збереження даних (Data-at-Rest) реалізовано концепцію вибіркового шифрування полів бази даних (Field-Level Encryption) за стандартом AES-256. Такий підхід гарантує, що навіть у випадку компрометації серверної інфраструктури, найбільш конфіденційні дані — назви ліків та схеми дозування — залишаться недоступними для зловмисників.

Аналітична складова системи, реалізована на базі Node.js та MySQL, дозволяє трансформувати сирі логи подій у цінні статистичні звіти. Впровадження авторського алгоритму розрахунку Індексу пунктуальності (PI) та використання автоматизованих Cron-завдань для моніторингу пропущених доз відкриває нові можливості для дистанційної діагностики. Система не просто фіксує події, а й аналізує динаміку відхилень від графіку, що дозволяє опікунам та медичному персоналу отримувати випереджаючі сповіщення про потенційні проблеми в терапевтичному процесі.

Мобільний застосунок, розроблений на фреймворку React Native, став сполучною ланкою між складною технічною інфраструктурою та кінцевим користувачем. Завдяки реалізації складних механізмів візуалізації аналітики, пацієнт отримує наочний зворотний зв'язок про свій прогрес, що значно підвищує його мотивацію та залученість у процес одужання. Застосування Axios Interceptors для управління JWT-токенами та впровадження нативних графічних бібліотек підкреслює професійний підхід до розробки користувацького інтерфейсу, що відповідає сучасним стандартам UX/UI у сфері охорони здоров'я.

Перспективи подальшого розвитку та вдосконалення системи вбачаються у розширенні її функціональних можливостей шляхом інтеграції методів машинного навчання (Machine Learning) для предиктивного аналізу стану пацієнта. Майбутні ітерації розробки можуть включати впровадження нейронних мереж для виявлення аномалій у поведінці користувача, що дозволить прогнозувати ризик повного припинення терапії на основі динаміки

зміни Індексу пунктуальності. Крім того, апаратна частина комплексу може бути модернізована шляхом інтеграції додаткових біометричних датчиків (пульсоксиметрів, тонометрів) для кореляції факту прийому ліків зі змінами вітальних показників організму в реальному часі. Важливим напрямком розширення є також перехід до децентралізованих методів зберігання даних на базі технології Blockchain для забезпечення абсолютної незмінності медичних записів та створення транскордонних систем медичного моніторингу. З точки зору апаратної реалізації, планується розробка енергонезалежного модуля на базі альтернативних джерел живлення та впровадження підтримки протоколу LoRaWAN для забезпечення зв'язку в умовах відсутності стабільного Wi-Fi покриття, що зробить систему придатною для використання в сільській місцевості або під час подорожей.

Таким чином, розроблений комплекс є не просто технічним засобом, а повноцінною екосистемою, яка інтегрує передові методи інформаційної безпеки, автоматизації та аналізу даних. Впровадження такої системи в медичну практику має потенціал не лише оптимізувати роботу медичного персоналу, а й суттєво покращити якість життя мільйонів пацієнтів, забезпечуючи їм впевненість у безпеці та своєчасності лікування. Результати, отримані під час практики, підтверджують повну готовність архітектурних рішень до подальшої промислової реалізації та масштабування.

ПЕРЕЛІК ПОСИЛАНЬ

1. New survey reveals $2 trillion market opportunity for cybersecurity technology and Service Providers / B. Aiyer et al. *McKinsey & Company*. 2022. URL: https://www.mckinsey.com/capabilities/risk-and-resilience/our-insights/cybersecurity/ (дата звернення: 19.12.2025).

2. Internet of Medical Things (IoMT) market size, share & trends analysis report by product, by services, by application, by end-use, and segment forecasts, 2023–2030. Grand View Research. 2023. 110 p.

3. Al-Garadi M. A. A survey of machine and deep learning methods for internet of things (IoT) security / M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali. // IEEE Communications Surveys & Tutorials. 2020. Vol. 22, no. 3. P. 1646–1685.

4. Security and privacy in the internet of medical things: taxonomy, security requirements, and future challenges / S. J. Hussain et al. // IEEE Access. 2021. Vol. 9. P. 114311–114334.

5. Ray P. P. A survey on internet of things signals for health care / P. P. Ray, M. Mukherjee, K. A. G. Shu. // IEEE Transactions on Industrial Informatics. 2021. Vol. 17, no. 5. P. 3014–3026.

6. MQTT Version 5.0. OASIS Standard. 2019. URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html (дата звернення: 19.12.2025).

7. ESP32 Series Datasheet. Espressif Systems. 2023. URL: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (дата звернення: 19.12.2025).

8. Barka E. Performance analysis of lightweight encryption algorithms for IoT health monitoring / E. Barka, C. A. Kerrache. // Journal of Medical Systems. 2021. Vol. 45, no. 5. P. 56.

9. NIST Selects ‘Ascon’ Family for Lightweight Cryptography Standard. National Institute of Standards and Technology. 2023. URL:

https://www.nist.gov/news-events/news/2023/02/nist-selects-ascon-family-lightweight-cryptography-standard (дата звернення: 19.12.2025).

10. Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446. 2018. URL: https://datatracker.ietf.org/doc/html/rfc8446 (дата звернення: 19.12.2025).

11. Tilkov S. Node.js: Using JavaScript to Build High-Performance Network Programs / S. Tilkov, S. Vinoski. // IEEE Internet Computing. 2010. Vol. 14, no. 6. P. 80–83.

12. React Native Documentation / Meta Platforms Inc. 2023. URL: https://reactnative.dev/docs/getting-started (дата звернення: 19.12.2025).

13. Health Insurance Portability and Accountability Act of 1996 (HIPAA). Public Law 104-191. 104th Congress. 1996.

14. Regulation (EU) 2016/679 (General Data Protection Regulation). Official Journal of the European Union. 2016. L 119. P. 1–88.

15. Banks A. MQTT Version 3.1.1 / A. Banks, R. Gupta. OASIS Standard. 2014. URL: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html (дата звернення: 19.12.2025).

16. Hoffstein J. An Introduction to Mathematical Cryptography / J. Hoffstein, J. Pipher, J. H. Silverman. Springer, 2014. 550 p.

17. MySQL 8.0 Reference Manual. Oracle Corporation. 2023. URL: https://dev.mysql.com/doc/refman/8.0/en/ (дата звернення: 19.12.2025).

18. Sequelize ORM Documentation. 2023. URL: https://sequelize.org/docs/v6/ (дата звернення: 19.12.2025).

19. Nodemailer: Send emails from Node.js. 2024. URL: https://nodemailer.com/ (дата звернення: 19.12.2025).

20. Koblitz N. Elliptic Curve Cryptography / N. Koblitz, A. Menezes, S. Vanstone. // Designs, Codes and Cryptography. 2000. Vol. 19. P. 173–193.