Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп'ютерних наук і технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти − бакалавр
за освітньо-професійною програмою
«Комп'ютерні науки»
зі спеціальності
122 – Комп'ютерні науки

тема роботи:

«Розробка веб-системи цифрового супроводу реабілітації та медичного контролю пацієнтів»

Виконав студент гр. КН-22-1 _____ Воліков Д.Е.

Керівник _____ Жосан А.А.

Нормоконтроль _____ Маринич І. А.

Завідувач кафедри _____ Рубан С. А.

Кривий Ріг – 2026

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет: *інформаційних технологій*

Кафедра: *автоматизації, комп'ютерних наук і технологій*

Ступінь вищої освіти: *Бакалавр*

Спеціальність: *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ

Зав. кафедрою: *к.т.н. Рубан С.А.*

_____

« *27* » *березня* 2026 р.

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

студентові групи КН-22-1 Волікову Д.Е.

1. Тема кваліфікаційної роботи: *Розробка веб-системи цифрового супроводу реабілітації та медичного контролю пацієнтів*
затверджено наказом по університету № *173с* від *30.03.2026 р.*

2. Термін здачі кваліфікаційної роботи: *15.06.2026 р.*

3. Склад кваліфікаційної роботи: *Пояснювальна записка обсягом 79 с., презентація у Microsoft PowerPoint в електронному та друкованому вигляді*

4. Консультанти кваліфікаційної роботи:

Розділ 1-2 доц. Рубан С.А.

Нормоконтроль доц. Маринич І. А.

5. Календарний план:

№	Етапи роботи	Термін виконання
1	*Вступ*	*01.04.26*
2	*Розділ 1*	*05.04.26*
3	*Розділ 2*	*01.05.26*
4	*Висновки*	*25.05.26*
5	*Оформлення кваліфікаційної роботи*	*30.05.26*
6	*Підготовка презентації та графічного матеріалу*	*10.06.26*
7	*Підготовка доповіді до захисту*	*15.06.26*

6. Дата видачі завдання: *28.01.2026р.*

Керівник _____ Жосан А.А.

7. Запевнення: *Я,* Воліков Д.Е., *запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.*

Із чинним Положенням про академічну доброчесність Криворізького національного університету ознайомлений.

Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі умисних порушень робота не допускається до захисту або оцінюється незадовільно.

Студент _____ Воліков Д.Е.

АНОТАЦІЯ

Кваліфікаційна робота «Розробка веб-системи цифрового супроводу реабілітації та медичного контролю пацієнтів» на здобуття ступеня вищої освіти – бакалавр, за спеціальністю 122 Комп'ютерні науки., Кривий Ріг, Криворізький національний університет, 2026, 79 с.

Актуальність теми полягає в розробці веб-системи призначеної для ефективного безперервного контролю за виконанням пацієнтами призначеного плану лікування та своєчасного реагування на можливі відхилення, які потребують довготривалої фізичної та психологічної реабілітації, а також необхідність оптимального розподілу ресурсів медичних закладів в умовах повномасштабної війни.

Мета роботи є розробка веб-системи цифрового супроводу реабілітації та медичного контролю пацієнтів, яка забезпечує дистанційний моніторинг стану пацієнта, підтримку виконання реабілітаційних заходів та зниження навантаження на медичний персонал за рахунок автоматизації рутинних процесів

У першому розділі було проведено аналіз сучасних інформаційних систем у сфері охорони здоров'я показав, що більшість існуючих рішень орієнтовані або на ведення медичної документації, або на дистанційні консультації, тоді як питання контролю виконання індивідуальних реабілітаційних програм, моніторингу стану пацієнта та своєчасного інформування лікаря потребують подальшого розвитку.

На основі проведеного аналізу спроєктовано структуру бази даних, визначено взаємозв'язки між основними сутностями системи та розроблено архітектуру програмного рішення з використанням сучасних вебтехнологій.

У другому розділі реалізовано веб-систему, побудовану за клієнт-серверною архітектурою з використанням React, TypeScript, Supabase та PostgreSQL. Реалізовано механізми автентифікації та розмежування доступу між ролями лікаря і пацієнта, що забезпечує безпечну роботу користувачів у межах власних функціональних можливостей.

КЛЮЧОВІ СЛОВА: РЕАБІЛІТАЦІЯ, МОНІТОРІНГ, ДИСТАНЦІЙНЕ КЕРУВАННЯ, МЕДИЧНИЙ КОНТРОЛЬ, REACT, WEB-ТЕХНОЛОГІЇ, IT, ВЕБСИСТЕМА.

ЗМІСТ

ВСТУП

Сучасний розвиток інформаційних технологій суттєво трансформує сферу охорони здоров’я, сприяючи переходу від традиційних моделей надання медичних послуг до цифрових систем, орієнтованих на безперервний моніторинг, автоматизацію рутинних процесів та підвищення якості медичного супроводу пацієнтів. Одним із найбільш актуальних напрямів цифрової медицини є впровадження веб-орієнтованих платформ, що забезпечують дистанційний контроль стану пацієнта, підтримку процесів реабілітації та ефективну взаємодію між пацієнтом і медичним персоналом.

Особливої актуальності дана проблема набуває в сучасних умовах функціонування системи охорони здоров’я України, яка перебуває під значним навантаженням через наслідки повномасштабної війни, збільшення кількості пацієнтів, які потребують довготривалої фізичної та психологічної реабілітації, а також необхідність оптимального розподілу ресурсів медичних закладів [1]. За таких умов традиційна модель періодичних очних консультацій не завжди дозволяє забезпечити ефективний безперервний контроль за виконанням пацієнтом призначеного плану лікування та своєчасне реагування на можливі відхилення.

Важливою проблемою сучасної медицини залишається недостатній рівень комплаєнсу пацієнтів, тобто дотримання ними рекомендацій лікаря щодо прийому медикаментів, виконання реабілітаційних вправ та регулярного контролю фізіологічних показників. Недостатній контроль за цими процесами може призводити до зниження ефективності лікування, подовження термінів відновлення та підвищення ризику повторних ускладнень [2].

Одним із перспективних шляхів вирішення зазначених проблем є створення інтегрованих цифрових систем, які поєднують функції медичного моніторингу, управління планом реабілітації, автоматизованих нагадувань та контролю виконання пацієнтом лікарських призначень. Саме до такого класу рішень

належить запропонована в межах даної кваліфікаційної роботи веб-система MedAlert.

Метою кваліфікаційної роботи є розробка веб-системи цифрового супроводу реабілітації та медичного контролю пацієнтів, яка забезпечує дистанційний моніторинг стану пацієнта, підтримку виконання реабілітаційних заходів та зниження навантаження на медичний персонал за рахунок автоматизації рутинних процесів.

Для досягнення поставленої мети необхідно вирішити такі задачі:

– проаналізувати сучасний стан цифрових рішень у сфері медичного моніторингу та реабілітації;

– виконати порівняльний аналіз існуючих інформаційних систем та визначити їх функціональні обмеження;

– дослідити міжнародні клінічні та технічні стандарти цифрової медицини;

– обґрунтувати вибір архітектурних рішень і технологічного стеку для реалізації системи;

– спроєктувати структуру бази даних та логіку взаємодії користувачів системи;

– реалізувати програмний прототип веб-системи;

– провести тестування функціональних можливостей розробленого програмного продукту.

Об'єктом дослідження є процес цифрового супроводу пацієнтів у межах реабілітаційного лікування та медичного моніторингу.

Предметом дослідження є методи, моделі та програмні засоби розробки веб-систем для автоматизації процесів реабілітаційного супроводу та дистанційного медичного контролю.

Для досягнення поставленої мети у роботі використано такі методи дослідження: аналіз наукової та технічної літератури, порівняльний аналіз існуючих програмних рішень, методи системного аналізу, методи проєктування інформаційних систем, моделювання структури баз даних, а також сучасні веб-технології програмної реалізації.

Практичне значення отриманих результатів полягає у можливості використання розробленої системи в медичних закладах, реабілітаційних центрах та приватній медичній практиці для підвищення ефективності супроводу пацієнтів, оптимізації роботи лікаря та покращення якості медичних послуг.

Структурно кваліфікаційна робота складається зі вступу, декількох розділів, висновків, списку використаних джерел та додатків.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Актуальність впровадження систем адаптивного моніторингу та дистанційного керування процесами реабілітації

Сучасний етап розвитку систем охорони здоров’я характеризується активною цифровою трансформацією медичних процесів, що зумовлено зростанням обсягів медичних даних, підвищенням потреби у доступності медичних послуг та необхідністю оптимізації взаємодії між пацієнтом і медичним працівником. Одним із ключових напрямів такого розвитку є впровадження цифрових інформаційних систем, орієнтованих на забезпечення безперервного супроводу пацієнтів, автоматизацію процесів моніторингу стану здоров’я та підвищення ефективності реабілітаційних заходів [1].

Особливої актуальності зазначене питання набуває в умовах сучасних соціально-медичних викликів, пов’язаних із наслідками повномасштабної війни в Україні. За даними World Health Organization, кількість осіб, які потребують довготривалої фізичної та психологічної реабілітації внаслідок бойових дій, щороку зростає, що створює додаткове навантаження на медичну систему країни [2]. До категорії таких пацієнтів належать не лише військовослужбовці, а й цивільне населення, яке зазнало травм, оперативних втручань або потребує тривалого відновлення після тяжких захворювань.

Традиційна модель медичного супроводу, заснована виключно на періодичних очних консультаціях, демонструє низку обмежень. По-перше, вона не забезпечує постійного контролю за дотриманням пацієнтом призначеного плану лікування або реабілітації. По-друге, лікар не має можливості оперативно відслідковувати зміни стану пацієнта між візитами. По-третє, значна частина рутинних дій, пов’язаних із нагадуваннями, збором даних та оцінкою динаміки, виконується вручну, що збільшує навантаження на медичний персонал [3].

У міжнародній медичній практиці зазначена проблема розглядається через концепцію безперервного супроводу пацієнта (continuous patient care), яка передбачає інтеграцію цифрових технологій у процес лікування та реабілітації. Основною ідеєю такого підходу є перенесення частини контрольних функцій із клінічного середовища у цифровий простір, де пацієнт може щоденно вносити показники стану здоров'я, фіксувати виконання призначень та отримувати автоматизовані нагадування, а лікар - дистанційно аналізувати динаміку лікування та приймати обґрунтовані клінічні рішення [4].

Окремої уваги потребує питання комплаєнсу пацієнта (patient compliance), тобто ступеня дотримання ним призначених рекомендацій. За даними World Health Organization, у середньому лише близько 50% пацієнтів із хронічними захворюваннями повністю дотримуються лікувальних протоколів, що суттєво знижує ефективність терапії та збільшує ризик ускладнень [5]. У контексті реабілітації цей показник є ще більш критичним, оскільки пропуск фізичних вправ, прийому медикаментів або контрольних вимірювань безпосередньо впливає на швидкість та якість відновлення.

Впровадження адаптивних веб-систем цифрового супроводу дозволяє частково вирішити зазначені проблеми шляхом автоматизації ключових процесів: формування персоналізованих планів реабілітації, організації системи нагадувань, збору медичних показників, оцінки виконання призначень та виявлення ризикових відхилень. Важливою перевагою таких рішень є їх доступність через браузер без необхідності встановлення окремого програмного забезпечення, що суттєво розширює потенційне коло користувачів [6].

Таким чином, розробка веб-системи цифрового супроводу реабілітації та медичного контролю пацієнтів є актуальним і соціально значущим завданням, спрямованим на підвищення ефективності реабілітаційних процесів, зменшення навантаження на медичний персонал та забезпечення якісного безперервного моніторингу стану пацієнтів.

1.2 Аналіз сучасних інформаційних систем медичного моніторингу та цифрової реабілітації

Розвиток цифрової медицини у світі сприяв появі значної кількості інформаційних систем, орієнтованих на автоматизацію окремих етапів лікувального процесу, дистанційний моніторинг пацієнтів та підтримку прийняття клінічних рішень. Водночас більшість існуючих рішень зосереджені на окремих функціональних аспектах - електронному документообігу, телемедичних консультаціях, контролі медикаментозної терапії або дистанційній реабілітації - і лише частково вирішують задачу комплексного цифрового супроводу пацієнта.

У межах даного дослідження доцільним є аналіз найбільш репрезентативних сучасних платформ, що використовуються в Україні та за кордоном.

1.2.1 Національна електронна система охорони здоров'я україни (міс «helsi.me»)

Електронна медична система «Helsi.me» є найбільшим централізованим сервісом цифрової медицини в Україні, інтегрованим із Центральною базою даних eHealth (ЕСОЗ). З інженерної точки зору архітектура системи побудована як хмарне монолітне рішення, що забезпечує взаємодію між пацієнтами, лікарями первинної та вторинної ланок, а також державними органами контролю (НСЗУ).

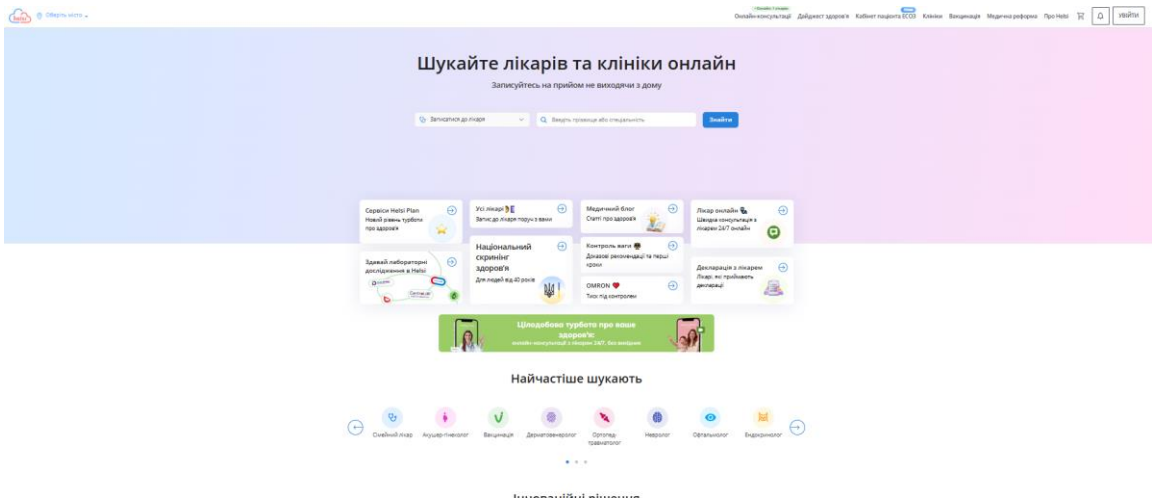


Рисунок 1.1 – Платформа Helsi.me

Основними компонентами системи є «Кабінет лікаря» та «Кабінет пацієнта» [7].

Основними функціональними можливостями системи є:

– онлайн-запис до лікаря;
– доступ до електронних медичних записів;
– перегляд електронних рецептів та направлень;
– ведення базових медичних даних пацієнта.

Для забезпечення безпеки даних використовується дата-центр із сертифікатом Комплексної системи захисту інформації (КСЗІ), що повністю відповідає нормативному полю України.

Незважаючи на масштабованість, система виконує переважно адміністративно-реєстраційну роль. Вона фіксує медичні події постфактум (дискретно, під час візиту). Водночас аналіз функціоналу показує, що система не підтримує:

– щоденний моніторинг виконання реабілітаційного плану;
– контроль комплаєнсу пацієнта;
– автоматизовану систему нагадувань щодо виконання призначень;
– алгоритми адаптивного реагування на зміни стану пацієнта.

Як показано на рисунку 1.2, у системі Helsi інформаційні потоки мають лінійний та епізодичний характер, де центральним вузлом є державна реєстраційна база даних.

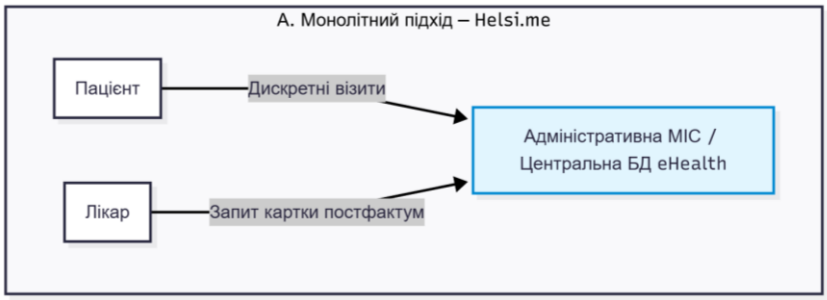


Рисунок 1.2 – Схема організації інформаційної взаємодії суб'єктів у МІС

Отже, Helsi орієнтована переважно на адміністративний супровід медичних послуг, а не на безперервний процес реабілітації.

1.2.2 Аналіз міжнародних цифрових медичних систем

У контексті систем безперервного супроводу пацієнтів більш прогресивним є досвід країн Європейського Союзу.

Зокрема, ***Естонія*** є світовим лідером у сфері e-Governance, і її національна система e-Health (яка наприкінці 2023 року пройшла масштабну модернізацію та трансформувалася з платформи Digilugu на інтегрований «**Terviseportaal**») є еталоном децентралізованої архітектури [8]. Система функціонує в межах безпечного державного середовища обміну даними X-Road. Основний принцип архітектури — інтероперабельність: будь-яка приватна або державна клініка підключає свої локальні бази даних до єдиного реєстру через стандартизовані шифровані шлюзи.

Портал пацієнта (рис 1.3) реалізує концепцію «часової шкали» (Timeline), де всі медичні події, аналізи, операції та вакцинації об'єднані в один логічний ланцюжок. Пацієнт має повний контроль над своїми даними: він може бачити лог-файли (хто саме і коли переглядав його картку) та закривати доступ до певних документів.

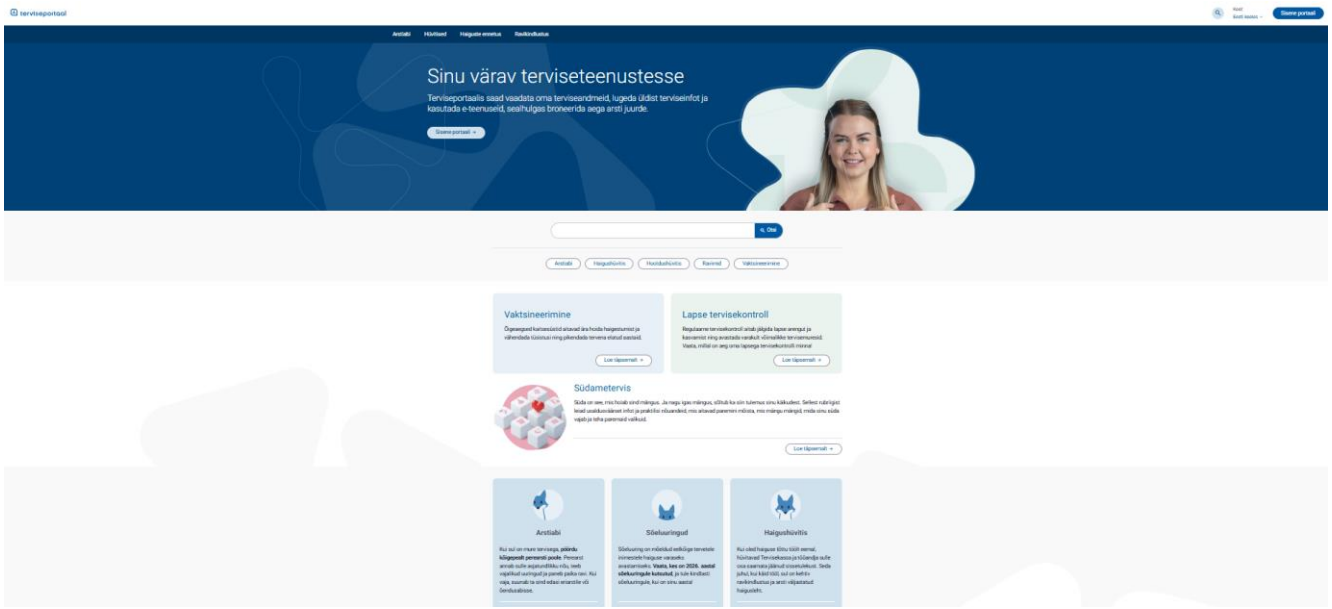


Рисунок 1.3 - Національна система e-Health Естонії «Terviseportaal»

Її перевагами є:

- централізоване зберігання медичних даних;
- електронні рецепти;
- міжсистемна інтероперабельність;
- доступ пацієнта до власної медичної історії.

Платформа розроблена як макросистема для збереження глобальних медичних документів (епікризів, рецептів). Вона не містить гнучких мікро-сервісів для оперативного керування специфічними планами відновлення (наприклад, покрокових комплексів вправ під керівництвом конкретного реабілітолога). Вона орієнтована на стандартизовані державні медичні протоколи, що ускладнює її швидку адаптацію та персоналізацію під щоденні потреби пацієнта.

У 2020 році *Німеччина* першою у світі на законодавчому рівні (Закон про цифрове забезпечення — DVG) впровадила поняття DiGA (Digitale Gesundheitsanwendungen) — «додатки за рецептом» [9]. Це програмне забезпечення (мобільні або веб-додатки), які пройшли сертифікацію Федерального інституту ліків і медичних виробів (BfArM) як медичні вироби з доведеною клінічною ефективністю. Лікар виписує пацієнту рецепт на додаток, а страхова компанія повністю покриває його вартість (в середньому близько 300–500 євро за 3 місяці курсу).

У реєстрі DiGA станом на 2026 рік перебувають десятки додатків, зокрема спеціалізовані платформи для реабілітації опорно-рухового апарату (наприклад, Vivira або Kaia Rückenschmerzen) [10]. Архітектура таких систем базується на зборі метрик пацієнта (Patient-Reported Outcome Measures, PROMs) та динамічному коригуванні щоденних відеотренувань на основі зворотного зв'язку про рівень болю. Згідно з оновленими стандартами DiGAV від 2026 року, виробники зобов'язані щоквартально генерувати агреговані знеособлені дані про задоволеність пацієнтів та тривалість використання додатків.

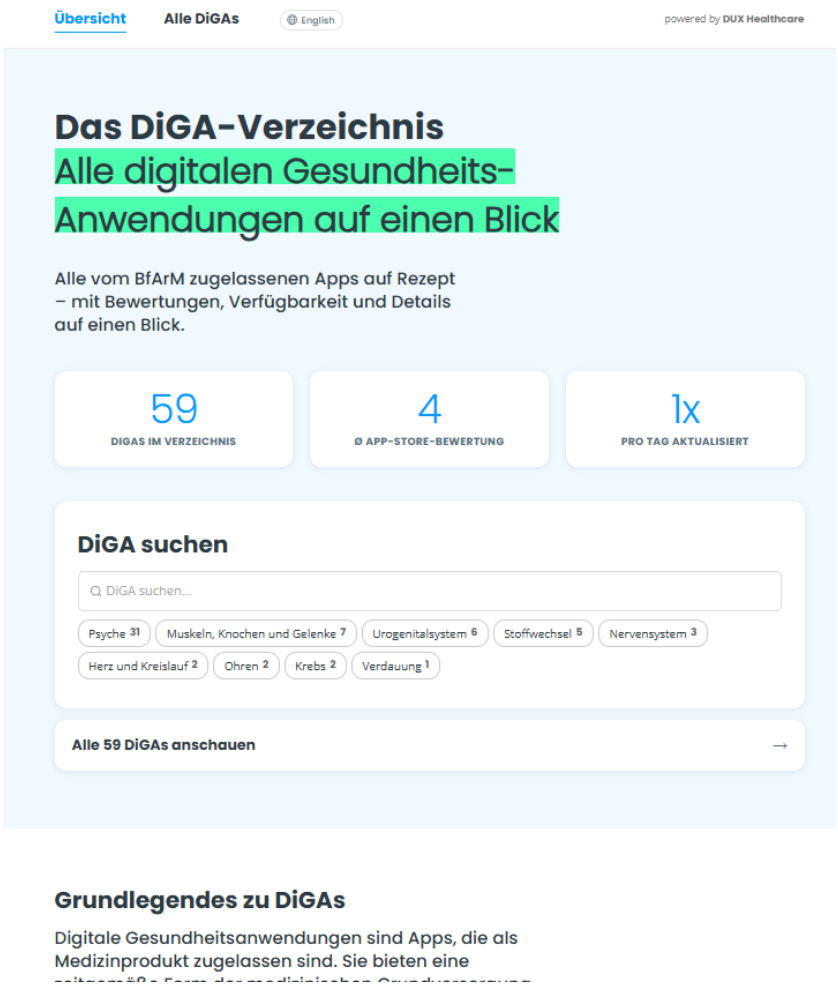


Рисунок 1.4 - Довідник DiGA: Всі програми цифрової охорони здоров'я

Переваги DiGA:

- медична сертифікація;
- доказова ефективність;
- орієнтація на self-management пацієнта.

Головним архітектурним обмеженням DiGA є їхня комерційна закритість та жорстка ізольованість. Кожен додаток створюється окремою компанією-розробником під конкретну нозологію (наприклад, лише для болю в спині). Це створює проблему фрагментації даних: лікар не має єдиної консолідованої панелі (Clinical Dashboard), де він міг би бачити загальний статус усіх своїх пацієнтів із різними патологіями, що суттєво перевантажує робочі процеси медичного персоналу.

1.2.3 Аналіз систем цифрової реабілітації

Окремим класом у структурі сучасного ринку медичного програмного забезпечення є вузькоспеціалізовані комерційні платформи телереабілітації, яскравим представником яких є європейська система «ReHub», розроблена компанією Bio-sensing Solutions SL (DyCare, Іспанія) [11]. На відміну від універсальних інформаційних систем, дана веб-платформа позиціонується як сертифікований медичний виріб (CE Medical Device), орієнтований виключно на фізичну реабілітацію та кінезотерапію пацієнтів.

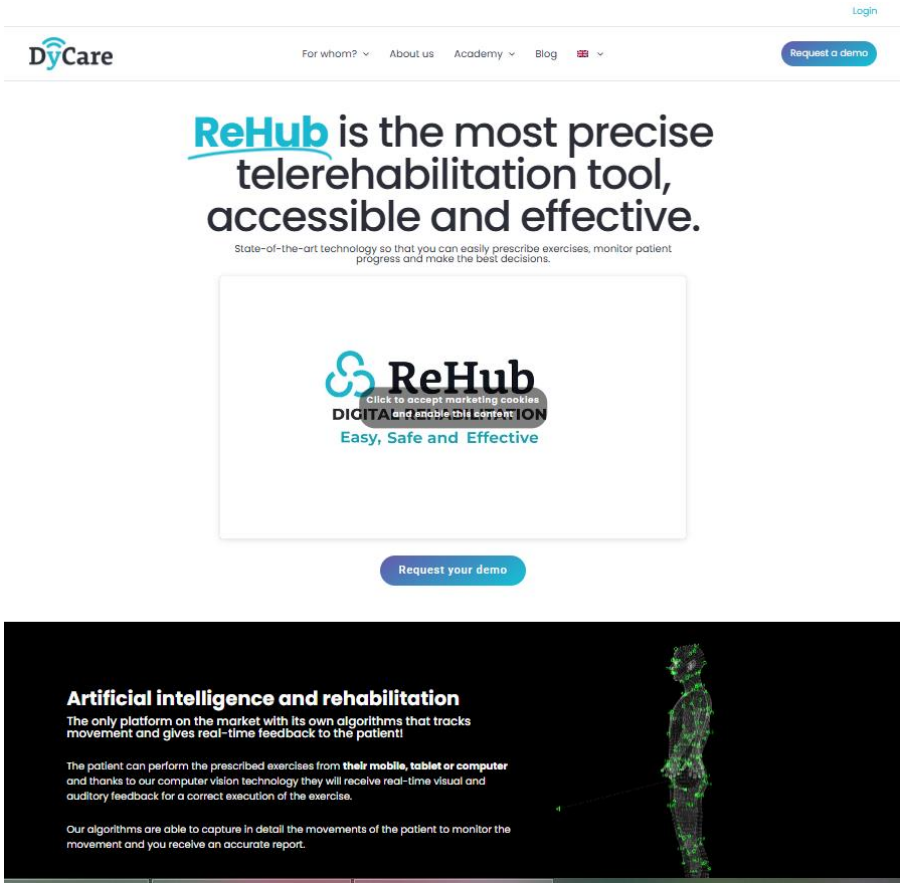


Рисунок 1.5 - Платформа телереабілітації «ReHub»

Архітектурною та технологічною особливістю «ReHub» є інтеграція алгоритмів штучного інтелекту та комп'ютерного зору (computer vision). Система дозволяє пацієнту виконувати призначені фізичні вправи перед камерою будь-якого смартфона, планшета або комп'ютера, при цьому програмні алгоритми відстежують біомеханіку рухів у реальному часі. Пацієнт отримує миттєвий

аудіо- та візуальний зворотний зв'язок (real-time feedback) щодо коректності виконання вправи, амплітуди рухів та темпу. Лікар, у свою чергу, отримує деталізований кількісний та якісний звіт (report metrics), на основі якого може гнучко коригувати терапію з бібліотеки, що налічує понад 2000 клінічно валідованих вправ. Платформа відповідає суворим вимогам безпеки даних (GDPR, ISO 13485, ISO 27001 та SOC 2) [11].

Попри високу технологічність у сфері розпізнавання рухів, «ReHub» має виражені архітектурні та функціональні обмеження у контексті комплексної медичної взаємодії:

Фокус виключно на ЛФК: Система повністю ізольована від загального терапевтичного контуру пацієнта - вона не містить модулів трекінгу медикаментозного комплаєнсу (прийому ліків), не підтримує ведення загального медичного базису (алергії, ІМТ, супутні патології) та не реалізує алгоритми клінічного триажу за фізіологічними показниками (такими як шкала NEWS2).

Продукт поширюється за пропрієтарною моделлю B2B (для госпіталів та страхових компаній), що унеможливлює його вільне та швидке впровадження у вітчизняну клінічну практику на рівні окремих лікарів чи невеликих реабілітаційних центрів.

1.2.4 Порівняльний аналіз існуючих рішень

Для систематизації результатів проведеного аналізу предметної області та чіткого визначення функціональних вимог до проектованої веб-системи, у таблиці 1.1 наведено порівняльну характеристику досліджених систем.

Таким чином, проведений порівняльний аналіз підтверджує, що існуючі рішення на ринку медичних ІТ є фрагментованими: великі національні платформи (Helsi, Terviseportaal) зосереджені на бюрократичних та реєстраційних функціях, а комерційні західні додатки (ReHub) — на вузьких завданнях кінезотерапії, ігноруючи системний моніторинг загального стану здоров'я та контроль медикаментозного комплаєнсу.

Таблиця 1.1 Порівняльний аналіз існуючих рішень

Функціональність	Helsi	Estonian e-Health	DiGA	ReHub
Електронна медична картка	+	+	-	-
Онлайн взаємодія лікар–пацієнт	+	+	±	+
Контроль реабілітації	-	-	±	+
Нагадування про призначення	-	-	+	±
Контроль комплаєнсу	-	-	±	±
Міжсистемна інтероперабельність	±	+	±	-
Адаптивний моніторинг	-	-	-	±
Збирання суб'єктивних метрик (PROMs)	-	-	±	+
Автоматизований клінічний триаж (NEWS2)	-	-	-	-

Це обґрунтовує необхідність розробки веб-системи цифрового супроводу реабілітації та медичного контролю пацієнтів, яка заповнить цей технологічний вакуум, поєднавши доступність веб-технологій PWA із гнучким інструментарієм клінічного оцінювання та інтерактивного планування.

1.3. Міжнародні клінічні та технічні стандарти цифрової медицини

Розробка сучасних інформаційних систем у сфері цифрової медицини потребує врахування міжнародно визнаних клінічних і технічних стандартів, які забезпечують коректність обробки медичних даних, підвищують якість прийняття рішень та створюють передумови для подальшої інтеграції інформаційних систем у національні та міжнародні цифрові екосистеми охорони здоров’я.

У контексті розробки веб-системи цифрового супроводу реабілітації та медичного контролю пацієнтів особливу увагу доцільно приділити трьом ключовим напрямам: використанню пацієнт-орієнтованих показників оцінки стану здоров’я, застосуванню стандартизованих клінічних шкал ризику та забезпеченню інтероперабельності медичних даних.

1.3.1 Концепція PROMs як основа пацієнт-орієнтованого моніторингу

У сучасній європейській медичній практиці ключовим вектором розвитку є перехід до пацієнт-орієнтованої моделі (Patient-Centered Care). Фундаментом цієї моделі є методологія PROMs (Patient-Reported Outcome Measures) — стандартизований збір даних про стан здоров'я, які надходять безпосередньо від пацієнта без інтерпретації чи коригування з боку лікаря або третіх осіб [12].

З технічної точки зору, інтеграція PROMs у веб-систему цифровізує суб'єктивні відчуття пацієнта, перетворюючи їх на структуровані дискретні або лінійні метрики. У процесі супроводу реабілітації PROMs реалізується через інтерактивні цифрові опитувальники, серед яких найважливішими є:

– візуально-аналогова шкала болю (ВАШ): Інструмент, де пацієнт оцінює інтенсивність болю від 0 (відсутність) до 10 (нестерпний біль). У веб-інтерфейсі це реалізується за допомогою динамічних повзунків (sliders).

– європейський опитувальник якості життя (EQ-5D-5L): Стандарт, що оцінює 5 компонентів (рухливість, самообслуговування, повсякденна активність, біль/дискомфорт, тривога/депресія) за 5-рівневою шкалою.

Розробка інтерфейсу користувача (UI) на базі React з урахуванням концепції PROMs вимагає створення адаптивних екранних форм із динамічною валідацією (client-side validation). Це дозволяє мінімізувати когнітивне навантаження на пацієнта та виключити помилки некоректного введення даних (user input errors), гарантуючи чистоту вибірки для подальшого аналізу лікарем.

У межах розроблюваної системи підхід PROMs будемо реалізовувати через:

– щоденне внесення пацієнтом суб’єктивних показників (біль, самопочуття, рівень активності);

– самооцінку виконання реабілітаційних завдань;

– фіксацію змін стану здоров’я в динаміці.

Таким чином, використання PROMs дозволить підвищити якість моніторингу та забезпечити пацієнт-орієнтований підхід до реабілітації.

1.3.2 Використання шкали NEWS2 для оцінки ризиків

Для автоматизації процесів дистанційного контролю та побудови алгоритмів триажу (сортування пацієнтів за ступенем тяжкості ризику) у систему закладається міжнародна шкала **NEWS2 (National Early Warning Score 2)**, розроблена Королівським коледжем лікарів Великої Британії (Royal College of Physicians) [13]. NEWS2 є визнаним стандартом для виявлення гострих погіршень стану пацієнта.

Математична логіка шкали базується на агрегації бальних оцінок шести ключових фізіологічних (біометричних) параметрів:

1. Частота дихання (циклів/хв).
2. Сатурація кисню (SpO_2, %).
3. Наявність або відсутність зовнішньої кисневої підтримки (Hypercapnic Respiratory Failure ризик).
4. Систолічний артеріальний тиск (мм рт. ст.).
5. Частота серцевих скорочень (пульс, уд./хв).
6. Рівень свідомості (шкала AVPU) або температура тіла (°C).

Кожному параметру залежно від ступеня його відхилення від фізіологічної норми присвоюється індекс від 0 до 3 балів. Сума цих балів формує інтегральний індекс NEWS2 (від 0 до 20).

Для проектованої інформаційної системи алгоритм Тріажу використовує тригерну логіку класифікації ризику:

0–4 бали: Низький клінічний ризик (зелений статус). Автоматичний запис даних у базу без генерації сповіщень лікарю.

5–6 балів АБО 3 бали за одним показником: Середній ризик (жовтий статус). Система маркує картку пацієнта в Clinical Dashboard та ініціює повторний замір через скорочений часовий інтервал.

7 і більше балів: Високий ризик (червоний статус). Система миттєво генерує асинхронну подію реального часу (Web Push Alert) для негайної реакції медичного персоналу.

У проектованій системі повна реалізація шкали NEWS2 не є обов’язковою, однак її концепція використовується як методологічна основа для побудови алгоритму автоматичного триажу, який аналізує критичні показники (наприклад, рівень болю, сатурацію, наявність задишки) та визначає пріоритетність реагування лікаря.

Це дозволяє реалізувати механізм раннього виявлення потенційно небезпечних станів пацієнта.

1.3.3 Стандарт HL7 FHIR як основа інтероперабельності медичних даних

Найважливішим архітектурним викликом при розробці баз даних для медичних інформаційних систем (МІС) є забезпечення інтероперабельності — здатності системи безшовно обмінюватися даними з іншими платформами (наприклад, національною eHealth/Helsi або європейськими EHR-реєстрами). Технологічним фундаментом для вирішення цього завдання є міжнародний стандарт *HL7 FHIR (Fast Healthcare Interoperability Resources)* [14].

Стандарт FHIR передбачає:

- уніфіковану структуру медичних ресурсів;
- використання REST API для обміну даними;
- підтримку форматів JSON та XML;
- масштабованість та гнучкість інтеграції.

У межах FHIR вся медична інформація декомпонується на атомарні інформаційні об'єкти — «ресурси» (Resources). Основою стандарту є такі базові ресурси:

- Patient — дані пацієнта;
- Practitioner — дані лікаря;
- Observation — результати медичних вимірювань;
- MedicationRequest — призначення медикаментів;
- CarePlan — план лікування або реабілітації.

Проектування структури реляційної бази даних (PostgreSQL у Supabase) із врахуванням FHIR-орієнтованих JSON-схем дозволяє закласти в систему високий

потенціал масштабованості. Це унеможливлює технологічну ізоляцію продукту та дозволяє у майбутньому підключати систему до будь-яких сертифікованих медичних шлюзів без реструктуризації ядра бази даних.

1.3.4 Нормативні вимоги до захисту медичної таємниці (GDPR)

Однією з ключових вимог до сучасних інформаційних систем у сфері охорони здоров'я є забезпечення конфіденційності, цілісності та захисту персональних медичних даних пацієнтів. Оскільки медична інформація належить до категорії чутливих персональних даних, її обробка повинна здійснюватися відповідно до міжнародних і національних нормативно-правових вимог [15].

Одним із базових міжнародних нормативних документів у сфері захисту персональних даних є General Data Protection Regulation (General Data Protection Regulation) — Регламент Європейського Союзу 2016/679, який визначає правила збору, зберігання, обробки та передачі персональних даних користувачів [16].

Основними принципами GDPR є:

- законність, справедливість та прозорість обробки даних;
- обмеження мети використання, тобто використання даних виключно для визначених медичних цілей;
- мінімізація даних, що передбачає збирання лише необхідної інформації;
- точність та актуальність даних;
- обмеження строків зберігання;
- забезпечення конфіденційності та безпеки.

У контексті медичних інформаційних систем особливо важливими є положення щодо:

- отримання *інформованої згоди* користувача на обробку персональних даних;
- права користувача на доступ до власних даних;
- можливості коригування або видалення інформації;
- обмеження доступу третіх осіб до медичних записів.

На національному рівні питання захисту медичної інформації в Україні регулюється Законом України Про захист персональних даних, а також нормами законодавства щодо медичної таємниці, визначеними в Основах законодавства України про охорону здоров’я [17].

У межах розроблюваної системи вимоги щодо захисту медичних даних враховуються на рівні архітектури системи через:

- механізм *автентифікації та авторизації користувачів*;
- *рольову модель доступу* (пацієнт, лікар, адміністратор);
- ізоляцію даних користувачів;
- використання захищених каналів передачі даних (*HTTPS*);
- зберігання даних у хмарному середовищі з підтримкою сучасних механізмів безпеки;
- обов’язкове підтвердження користувачем згоди на обробку персональних даних під час реєстрації.

Таким чином, дотримання міжнародних і національних вимог щодо захисту медичної таємниці є необхідною умовою проєктування сучасної цифрової медичної системи та безпосередньо впливає на довіру користувачів до програмного продукту.

1.4 Обґрунтування вибору архітектурних рішень та технологічного стеку

Ефективність функціонування розроблюваної веб-системи цифрового супроводу реабілітації безпосередньо залежить від архітектурної парадигми та вибору інструментальних засобів реалізації. Сучасні вимоги до медичних веб-додатків включають забезпечення високої швидкості відгуку інтерфейсу, кросплатформеність, здатність обробляти дані в реальному часі (Realtime data processing) та надійність збереження конфіденційної інформації. У межах даної роботи обґрунтовано доцільність використання трирівневої клієнт-серверної архітектури на базі сучасного стеку технологій: React, TypeScript, Supabase та концепції Progressive Web App (PWA).

Для реалізації клієнтської частини (Frontend) було обрано компонентно-орієнтовану бібліотеку **React** [18]. Вибір зумовлений використанням технології *Virtual DOM*, яка мінімізує витрати ресурсів на оновлення інтерфейсу користувача. Це критично для сторінки Clinical Dashboard лікаря, де графіки біометричних показників та черга триажу пацієнтів мають оновлюватися динамічно без перезавантаження всієї сторінки. Використання суворо типізованої мови **TypeScript** як надбудову над JavaScript, дозволяє виявити архітектурні та логічні помилки на етапі компіляції (compile-time errors). Для медичної системи, де тип даних (наприклад, передача числового значення пульсу чи рядкового статусу NEWS2) має критичне значення, використання TypeScript є інструментом забезпечення загальної стійкості до відмов (fault tolerance).

Як архітектурне рішення для забезпечення мобільності користувачів обрано концепцію **Progressive Web App (PWA)** [19]. Створення окремих нативних мобільних додатків під операційні системи Android (Kotlin/Java) та iOS (Swift) суттєво збільшує вартість та тривалість розробки, а також створює бар'єри для пацієнтів через необхідність завантаження програм із маркетів. Технологія PWA дозволяє трансформувати веб-сайт на базі React у кросплатформенний додаток, який користувач може встановити на екран смартфона безпосередньо з браузера. Завдяки використанню фонових скриптів — *Service Workers* — система отримує наступні переваги:

- можливість кешування статичних ресурсів для забезпечення автономної роботи (Offline Mode), що дозволяє пацієнту переглядати графік реабілітації навіть за відсутності мережевого з'єднання;
- інтеграція з *Push API* та *Notifications API* для реалізації фонових «розумних» нагадувань про прийом медикаментів чи фізіологічні заміри, коли сама вкладка браузера є закритою.

Для реалізації серверної частини (Backend) та бази даних використано хмарну платформу класу Backend-as-a-Service (BaaS) — **Supabase** [20]. Замість розгортання та підтримки власної громіздкої інфраструктури на базі Node.js чи Python, Supabase надає готовий набір інструментів, інтегрованих навколо

об'єктно-реляційної СУБД **PostgreSQL**. Вибір PostgreSQL зумовлений її високою надійністю, підтримкою складних реляційних зв'язків та вбудованим механізмом *Row Level Security (RLS)*, що було обґрунтовано в підрозділі 1.3, як один із механізмів забезпечення конфіденційності медичних даних.

Додатковою перевагою Supabase є вбудована технологія *Realtime-трансляції подій* через WebSockets. Будь-яка зміна в таблиці замірів (наприклад, якщо пацієнт вніс критичний показник тиску і спрацював тригер алгоритму NEWS2) миттєво, за частки секунди, передається на інтерфейс лікаря без постійних опитувань сервера (polling), що знижує навантаження на мережу та сприяє оперативному реагуванню медичного персоналу.

Таким чином, поєднання клієнтської JavaScript-бібліотеки React, реалізованої відповідно до концепції Progressive Web Application, з хмарним бекендом Supabase формує гнучку, масштабовану та безпечну архітектурну екосистему для побудови медичного веб-сервісу. Запропонований технологічний стек у поєднанні з ітераційною моделлю розробки забезпечує можливість поетапного створення, тестування та вдосконалення окремих функціональних модулів системи. Такий підхід дозволяє послідовно реалізувати базові компоненти (авторизацію, профілі користувачів, моніторинг показників), а в подальшому — масштабувати систему шляхом додавання функцій планування реабілітації, автоматизованих нагадувань та аналітичних модулів. Отже, обрані архітектурні рішення та технології повністю відповідають поставленим задачам проєктування веб-системи цифрового супроводу реабілітації та медичного контролю пацієнтів.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБ-СИСТЕМИ ЦИФРОВОГО СУПРОВОДУ РЕАБІЛІТАЦІЇ ТА МЕДИЧНОГО КОНТРОЛЮ ПАЦІЄНТІВ

2.1 Формалізація вимог до системи та моделювання бізнес-процесів

Проєктування будь-якої інформаційної системи розпочинається з формалізації предметної області, визначення основних учасників взаємодії та встановлення функціональних і нефункціональних вимог до програмного продукту. Для веб-системи цифрового супроводу реабілітації та медичного контролю пацієнтів такий етап є критично важливим, оскільки система повинна забезпечувати не лише зручність використання, але й підтримку безперервного медичного спостереження, автоматизацію рутинних процесів та підвищення ефективності взаємодії між лікарем і пацієнтом.

У межах даного дослідження було визначено три основні категорії користувачів системи: пацієнт, лікар та адміністратор. Концептуальна назва розроблюваного програмного рішення — **«MedAlert»**, що безпосередньо відображає інтелектуальну функцію оперативного клінічного сповіщення.

Пацієнт є кінцевим користувачем системи, який взаємодіє з платформою на щоденній основі. До його основних функцій належать: реєстрація в системі, заповнення первинної медичної анкети, перегляд індивідуального плану реабілітації, внесення щоденних фізіологічних показників, підтвердження виконання призначених завдань, перегляд історії лікування та отримання автоматичних нагадувань.

Лікар виступає ключовим суб’єктом керування лікувальним процесом. Його функції включають реєстрацію та верифікацію професійного профілю, формування та редагування індивідуальних планів реабілітації за допомогою модульного конструктора, оперативний моніторинг динаміки біометричних показників пацієнтів, аналіз розрахованого індексу комплаєнсу, обробку автоматизованих клінічних сповіщень та ведення цифрової карти пацієнта.

Адміністратор забезпечує системне управління платформою, контроль доступу, підтвердження лікарських профілів, аудит подій та підтримку безпеки інформаційного середовища.

На основі проведеного аналізу предметної області та визначених ролей користувачів було сформовано сукупність функціональних і нефункціональних вимог до програмної системи. Їх формалізація є необхідною передумовою подальшого проєктування архітектури програмного продукту, структури бази даних та алгоритмів функціонування системи.

Функціональні вимоги веб-системи:

– *реєстрація та автентифікація користувачів із підтримкою рольової моделі доступу.*

Система повинна забезпечувати можливість створення облікового запису нового користувача, автентифікацію за логіном і паролем або через зовнішнІ сервіси автентифікації (OAuth 2.), а також автоматичне визначення ролі користувача (пацієнт, лікар, адміністратор) з відповідним розмежуванням прав доступу до функціональних модулів.

– *ведення персональних профілів пацієнтів і лікарів*

Для кожного користувача має формуватися персональний профіль із можливістю перегляду, редагування та збереження персональних, медичних і професійних даних. Для лікаря передбачено процедуру верифікації, а для пацієнта — збереження медичного анамнезу та історії лікування.

– *створення, редагування та архівування індивідуальних планів реабілітації*

Лікар повинен мати можливість формувати персоналізований план лікування з використанням модульного конструктора, додавати необхідні етапи реабілітації, змінювати активні призначення та зберігати історію попередніх версій плану.

– *автоматичне формування щоденних завдань для пацієнта*

На основі сформованого лікарем плану система повинна автоматично генерувати перелік щоденних активностей пацієнта: прийом медикаментів, фізичні вправи, проведення вимірювань, здачу аналізів та планові візити.

– *внесення та збереження медичних показників у режимі реального часу*

Пацієнт повинен мати можливість вносити фізіологічні параметри (артеріальний тиск, пульс, температура, сатурація, рівень болю тощо), які автоматично зберігаються в базі даних та стають доступними лікарю для аналізу.

– генерація критичних сповіщень на основі визначених порогових значень

У разі перевищення допустимих медичних показників або пропуску критично важливих завдань система повинна автоматично формувати алерт для лікаря з відповідним рівнем пріоритету.

– автоматизований розрахунок рівня виконання плану лікування (індексу комплаєнсу - Compliance Score)

Система повинна аналізувати частоту та повноту виконання пацієнтом призначених завдань і обчислювати числовий показник комплаєнсу, який використовується для оцінки дисципліни пацієнта.

– підтримка модуля клінічного моніторингу та аналітики

Лікар повинен мати доступ до динамічних графіків, часових рядів медичних показників, статистики виконання плану та узагальнених аналітичних даних щодо пацієнтів.

– формування історії змін та журналу аудиту

Усі дії користувачів у системі (створення плану, редагування даних, виконання завдань, перегляд документів) повинні фіксуватися в журналі подій для забезпечення простежуваності змін.

– підтримка електронного документообігу

Система повинна забезпечувати можливість завантаження, зберігання, перегляду та керування медичними документами: результатами аналізів, виписками, зображеннями МРТ, рентгенограмами тощо.

– реалізація системи push-сповіщень (Push Notifications) та нагадувань

Пацієнт має отримувати автоматичні нагадування про необхідність виконання завдань, навіть у разі закритого браузера, завдяки використанню технології Progressive Web App.

– підтримка автономного режиму роботи (offline mode)

Система повинна забезпечувати можливість перегляду активного плану реабілітації та внесення даних без підключення до мережі з подальшою синхронізацією після відновлення інтернет-з’єднання.

До **нефункціональних вимог** віднесено:

– забезпечення конфіденційності персональних медичних даних відповідно до GDPR [14];

– високу масштабованість архітектури;

– кросплатформеність (desktop/mobile);

– підтримку двомовного інтерфейсу (українська/англійська);

– швидкодію системи в режимі реального часу;

– зручність інтерфейсу та адаптивний дизайн;

– відмовостійкість та резервування даних.

UML-моделювання прецедентів використання та динаміки бізнес-процесів

Узагальнена діаграма прецедентів для системи MedAlert включає такі сценарії:

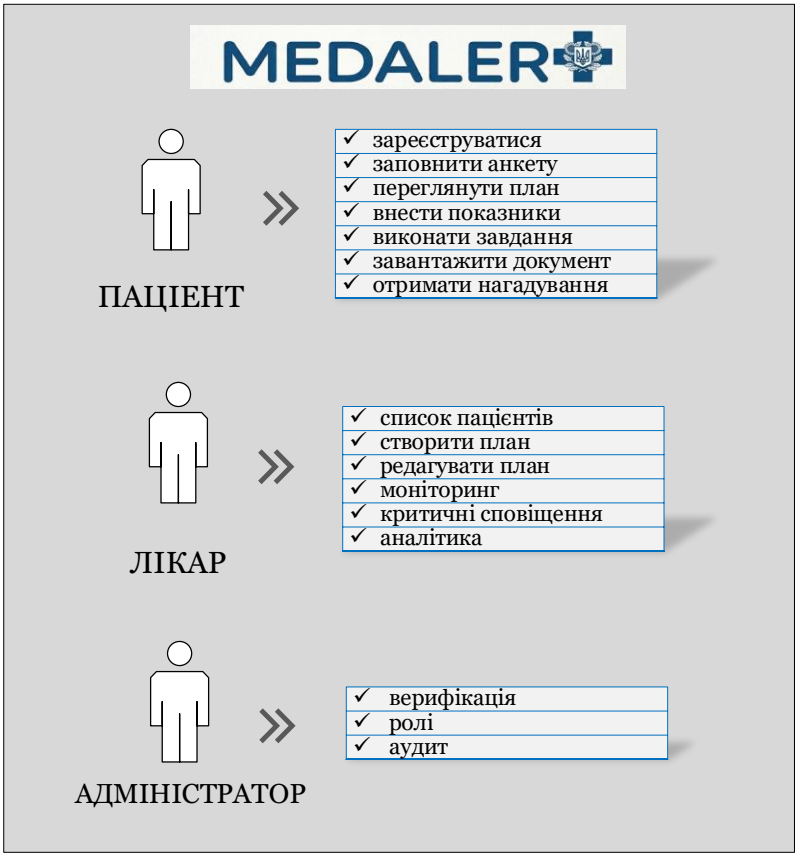


Рисунок 2.1 - Узагальнена діаграма прецедентів веб-системи MedAlert

Представлена UML-діаграма прецедентів демонструє розподіл функціональних обов’язків між основними ролями користувачів та відображає логіку взаємодії із ключовими модулями системи.

Також для опису послідовності виконання основного бізнес-процесу було сформовано **діаграму діяльності (Activity Diagram)** процесу цифрового супроводу реабілітації:

1. реєстрація пацієнта в системі;
2. заповнення первинної медичної анкети;
3. формування лікарем індивідуального плану реабілітації;
4. автоматичне генерування системою щоденних завдань;
5. виконання пацієнтом призначених активностей;
6. збереження отриманих даних у базі;
7. аналіз показників алгоритмом клінічного триажу;
8. формування критичного сповіщення (за необхідності);
9. прийняття рішення лікарем;
10. корекція плану лікування.

Таким чином, проведена формалізація вимог дозволила визначити ключові функціональні межі системи, описати взаємодію основних користувачів та сформувати основу для подальшого проєктування структури даних, алгоритмів функціонування та архітектури програмного рішення.

2.2. Проєктування та реалізація бази даних веб-системи «MedAlert»

Ефективність функціонування вебсистеми цифрового супроводу реабілітації «MedAlert» значною мірою залежить від правильно спроєктованої структури збереження даних. Оскільки система працює з персональними медичними даними, планами лікування, клінічними спостереженнями та журналами виконання реабілітаційних завдань, архітектура бази даних повинна забезпечувати цілісність інформації, контроль доступу до даних та можливість подальшого масштабування функціональності.

Для реалізації серверної частини системи було використано PostgreSQL у складі хмарної платформи Supabase. Даний підхід дозволив поєднати переваги класичної реляційної моделі даних із сучасними механізмами автентифікації користувачів, політиками безпеки Row Level Security (RLS), файловим сховищем та автоматичною генерацією API для взаємодії клієнтської частини із базою даних.

Архітектура інформаційної системи побудована за принципом логічного розділення користувацьких профілів, медичних даних пацієнтів, професійної інформації лікарів, індивідуальних реабілітаційних планів, клінічних записів, журналів виконання завдань, медичних документів та системних сповіщень.

2.2.1 Основні сутності системи та їх взаємозв'язки

Для забезпечення типізації бізнес-логіки та унеможливлення некоректного внесення записів на рівні СУБД було спроектовано та впроваджено чотири базові типи перерахувань (`ENUM`):

```
CREATE TYPE public.app_role AS ENUM ('patient', 'doctor');
CREATE TYPE public.task_log_status AS ENUM (
  'done',
  'skipped',
  'partial' );
CREATE TYPE public.care_plan_status AS ENUM (
  'active',
  'paused',
  'finished',
  'archived',
  'draft' );
CREATE TYPE public.care_module AS ENUM (
  'medication',
  'vitals',
  'pain',
  'exercise',
  'mobility',
  'wound',
  'lab',
  'visit',
  'symptoms',
  'mental',
 'nutrition'
 );
```

Базовою сутністю системи є таблиця `public.profiles`, яка містить загальну інформацію про всіх зареєстрованих користувачів незалежно від їх ролі. Вона виступає точкою розширення для вбудованого модуля автентифікації Supabase Auth (`auth.users`).

Таблиця містить такі основні атрибути:

```
profiles (
id UUID PRIMARY KEY,
email TEXT,
full_name TEXT,
phone TEXT,
role public.app_role,
created_at TIMESTAMPTZ,
updated_at TIMESTAMPTZ
)
```

`id` — унікальний ідентифікатор користувача (UUID);

`email` — електронна адреса;

`full_name` — повне ім'я користувача (прізвище, ім'я, по батькові);

`phone` — контактний номер телефону;

`role` — поточна роль користувача в системі;

`created_at` — дата створення запису;

`updated_at` — дата останнього оновлення.

Для реалізації суворої моделі ролей на рівні бази даних впроваджено системний перелік значень (`ENUM`) `app_role`:

```
CREATE TYPE public.app_role AS ENUM (
    'patient',
    'doctor'
);
```

Для реалізації рольової моделі додатково використовується таблиця `user_roles`, яка зберігає зв’язок між користувачем та його роллю:

```
user_roles (
  id UUID PRIMARY KEY,
  user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,
  role public.app_role,
  created_at TIMESTAMPTZ
)
```

Для розширення інформації про користувачів використовуються спеціалізовані таблиці:

- `patient_profiles`;
- `doctor_profiles`.

Таблиця `patient_profiles` містить персональні та медичні дані пацієнта:

```
patient_profiles (
    user_id UUID PRIMARY KEY,
    birth_date DATE,
    blood_type TEXT,
    rh_factor TEXT,
    height NUMERIC,
    weight NUMERIC,
    bmi NUMERIC,
    allergies TEXT,
    chronic_conditions TEXT,
    medications TEXT,
    surgeries TEXT,
    emergency_contact TEXT,
    patient_status TEXT
)
```

Наявність полів `height`, `weight` та `bmi` дозволяє зберігати антропометричні показники пацієнта, а поля `allergies`, `chronic_conditions`, `medications` і `surgeries` формують медичну частину анкети, необхідну лікарю для планування реабілітації.

Професійний профіль лікаря декомпоновано на кілька сутностей. Основна таблиця `doctor_profiles` містить базові кваліфікаційні характеристики:

```
doctor_profiles (
user_id UUID PRIMARY KEY,
gender TEXT,
avatar_url TEXT,
experience_years INTEGER,
qualification TEXT,
bio TEXT,
verification_status TEXT,
onboarded BOOLEAN, c
reated_at TIMESTAMPTZ,
updated_at TIMESTAMPTZ
)
```

Спеціалізації лікаря зберігаються окремо у таблиці `doctor_specializations`:

```
  doctor_specializations (
 id UUID PRIMARY KEY,
 doctor_id UUID,
 specialization TEXT,
 sub_specialization TEXT,
 created_at TIMESTAMPTZ
  )
```

Інформація про місце роботи лікаря винесена до таблиці `workplaces`:

```
 workplaces (
id UUID PRIMARY KEY,
doctor_id UUID,
institution_name TEXT,
institution_type TEXT,
department TEXT,
position TEXT,
work_phone TEXT,
email TEXT,
messenger TEXT,
created_at TIMESTAMPTZ
  )
```

Окремо можуть зберігатися професійні документи лікаря: ліцензії, дипломи та сертифікати. Для цього використовується таблиця `licenses` та приватне файлове сховище `doctor-documents`.

Така декомпозиція дозволяє уникнути дублювання даних, зберігати професійний профіль лікаря у нормалізованому вигляді та надалі розширювати систему верифікації медичних працівників.

2.2.2 Архітектура та модульний конструктор індивідуальних планів реабілітації

Центральним логічним елементом, що пов'язує лікаря та пацієнта в єдину екосистему, є сутність `care_plans`, яка реалізує механізм індивідуальних планів реабілітації. Структура таблиці підтримує повне версіонування та фіксацію станів призупинення.

Структура таблиці має вигляд:

```
CREATE TABLE public.care_plans (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
patient_id UUID NOT NULL REFERENCES public.profiles(id) ON DELETE RESTRICT,
doctor_id UUID NOT NULL REFERENCES public.profiles(id) ON DELETE RESTRICT,
title TEXT NOT NULL,
diagnosis TEXT,
start_date DATE NOT NULL,
end_date DATE,
status public.care_plan_status NOT NULL DEFAULT 'draft'::public.care_plan_status,
notes TEXT,
version INTEGER NOT NULL DEFAULT 1,
parent_plan_id UUID REFERENCES public.care_plans(id) ON DELETE SET NULL,
pause_reason TEXT,
paused_at TIMESTAMPTZ,
created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
updated_at TIMESTAMPTZ NOT NULL DEFAULT now()
);
```

Для відображення життєвого циклу плану використовується перелік статусів:

```
care_plan_status = (
'active',
'paused',
'finished',
'archived',
'draft'
)
```

Впровадження атрибута `version` та зв'язку `parent_plan_id` дозволяє лікарю модифікувати та зберігати нові ітерації лікувальних планів, не руйнуючи історичні дані попередніх етапів реабілітації. Поля `pause_reason` та `paused_at` фіксують параметри у разі тимчасового призупинення плану лікарем.

Кожен план зв'язаний одночасно з лікарем та пацієнтом через зовнішні ключі:

`patient_id → profiles.id`
`doctor_id → profiles.id`

Таким чином реалізується модель «один лікар — багато пацієнтів» та «один пацієнт — багато планів лікування».

Реалізація модульного конструктора реабілітаційних програм

Гнучкість та динамічне налаштування реабілітаційних програм забезпечується сутністю `public.care_plan_tasks`. Замість створення десятків статичних таблиць під кожен окремий вид терапії (медикаменти, ЛФК, аналізи), архітектура використовує гібридний реляційно-документний підхід за допомогою бінарних полів типу `JSONB`.

Кожне завдання належить певному плану лікування та містить набір параметрів конфігурації.

```
CREATE TABLE public.care_plan_tasks (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
care_plan_id UUID NOT NULL REFERENCES public.care_plans(id) ON DELETE CASCADE,
title TEXT NOT NULL,
module public.care_module NOT NULL,
mandatory BOOLEAN NOT NULL DEFAULT false,
order_index INTEGER NOT NULL DEFAULT 0,
config JSONB NOT NULL DEFAULT '{}'::jsonb,
schedule JSONB NOT NULL DEFAULT '{}'::jsonb,
created_at TIMESTAMPTZ NOT NULL DEFAULT now()
);
```

Особливістю реалізації є використання полів типу `config JSONB` та `schedule JSONB`, що дозволяє зберігати довільні неструктуровані параметри призначень без зміни схеми БД. Для забезпечення функціонування модульного конструктора, ці поля структуровані за чіткими схемами валідації на рівні бізнес-логіки.

Для класифікації завдань використовується перелік модулів:

```
care_module = (
'medication',   медикаменти
'vitals',       вітальні показники
'pain',         оцінка болю
'exercise',     реабілітаційні вправи
'mobility',     рухова активність
'wound',        контроль рани
'lab',          лабораторні аналізи
'visit',        візити
'symptoms',     симптоми
'mental',       психоемоційний стан
'nutrition'     харчування
 )
```

Зокрема, для модуля контролю вітальних показників (`module = 'vitals'`), атрибут `config JSONB` зберігає масив ідентифікаторів вимірювань, наприклад

```
  {
"measures": ["bp_sys", "bp_dia", "pulse", "saturation"]
  }.
```

Для модуля медикаментозної терапії (`module = 'medication'`), поле `config` містить параметри дозування:

```
  {
"dosage": "1 таблетка", "route": "per_os"
  }.
```

Поле `schedule JSONB` регламентує часову сітку виконання завдань, зберігаючи масив міток часу або періодичність у форматі

```
  {
"times": ["08:00", "20:00"], "daysOfWeek": [1, 2, 3, 4, 5]
  }.
```

Фактично дана структура реалізує конструктор планів лікування, в якому лікар може формувати індивідуальні програми шляхом комбінування різних модулів реабілітації.

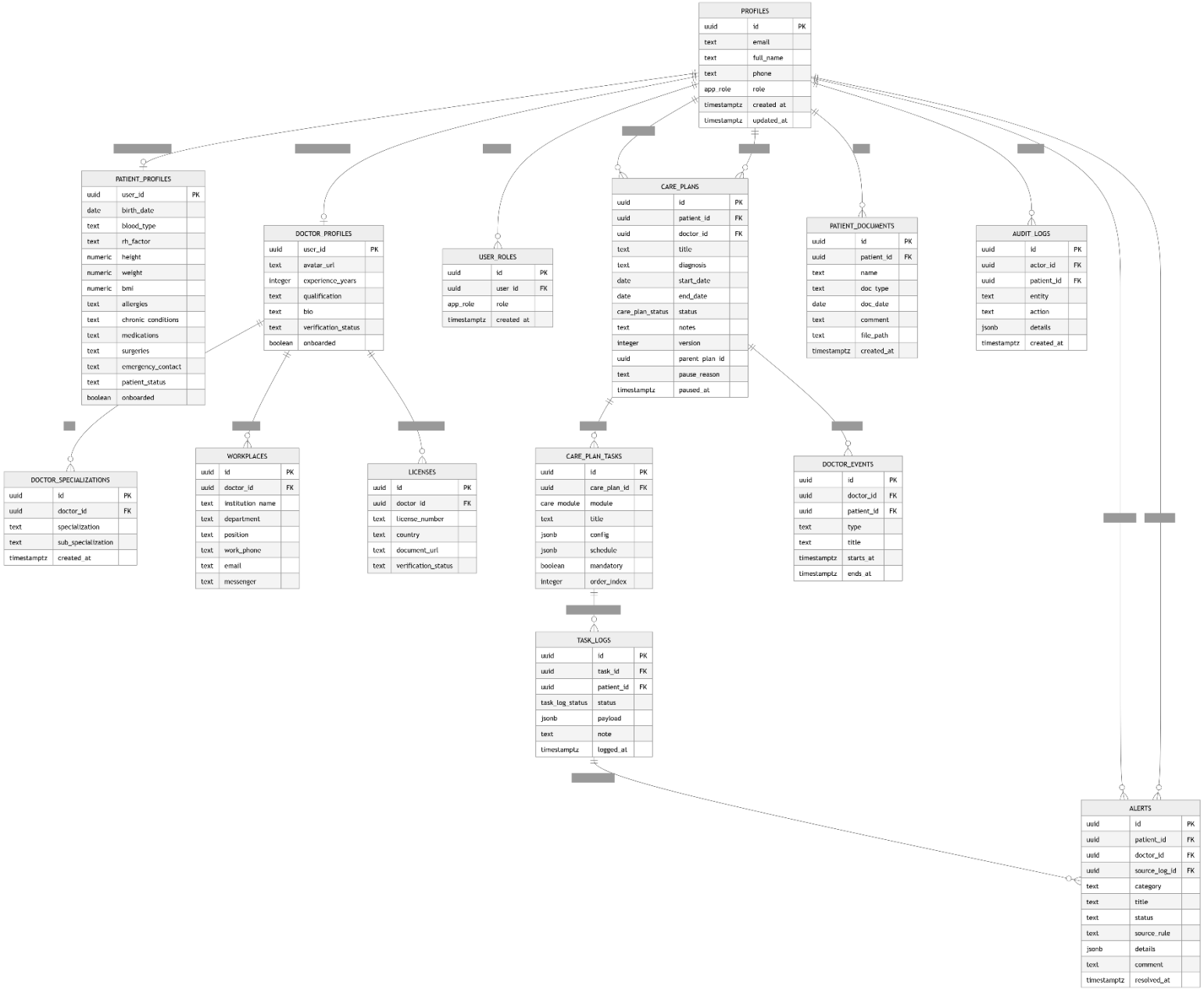


Рисунок 2.1 – ER-діаграма бази даних веб-системи «MedAlert».

2.2.3 Журнал виконання завдань, медичний моніторинг та аналітика комплаєнсу

Для фіксації результатів виконання завдань пацієнтом використовується таблиця `task_logs`.

Структура запису:

```
task_logs (
 id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
 task_id UUID NOT NULL REFERENCES public.care_plan_tasks(id) ON DELETE CASCADE,
 patient_id UUID NOT NULL REFERENCES public.profiles(id) ON DELETE RESTRICT,
 status public.task_log_status NOT NULL,
 payload JSONB NOT NULL DEFAULT '{}'::jsonb,
 note TEXT,
 logged_at TIMESTAMPTZ NOT NULL DEFAULT now()
);
```

Транзакційне поле `payload JSONB` динамічно адаптується під тип виконуваного модуля і використовується для збереження показників у структурованому форматі. При фіксації пацієнтом життєво важливих показників через діалогове вікно введення даних, у поле записується денормалізований об'єкт із ключами вимірювань та числовими значеннями: `{"bp_sys": 125, "bp_dia": 82, "pulse": 74}`. У випадку фіксації суб'єктивного рівня болю за модулем `pain`, об'єкт набуває вигляду `{"score": 7}`. Такий підхід мінімізує кількість операцій з'єднання (JOIN) при побудові часових рядів для аналітичних графіків лікаря. Для запобігання аномаліям дублювання транзакцій (коли пацієнт випадково або повторно вносить дані в межах однієї доби), на рівні бізнес-логіки обробки запитів реалізовано фільтр валідації ідемпотентності, який блокує створення дублюючих рядків для зв'язки `(patient_id, task_id, CURRENT_DATE)`.

Для відображення стану виконання використовується перелік:

```
task_log_status = (
 'done',            виконано
 'partial',         частково виконано
 'skipped'          пропущено
)
```

На основі цих даних реалізовано модуль аналітики комплаєнсу пацієнта (Compliance Score), який оцінює рівень дотримання рекомендацій лікаря.

Розрахунок виконується за формулою:

$$\text{Compliance Score} = \frac{\text{Done}+0.5*\text{Partial}}{\text{TotalTasks}} * 100\% \quad (2.1)$$

де:

Done — кількість виконаних завдань;

Partial — кількість частково виконаних завдань;

TotalTasks — загальна кількість зареєстрованих завдань на обраний період.

Програмна реалізація розрахунку показника представлена функцією

```
 calcCompliance:
export function calcCompliance(logs: ComplianceLog[]): number {
if (!logs.length) return 0;
const done = logs.filter((l) => l.status === "done").length;
const partial = logs.filter((l) => l.status === "partial").length;
return Math.round(((done + 0.5 * partial) / logs.length) * 100);
  }
```

Якщо TotalTasks = 0 (початок курсу, відсутність записів у журналі), система повертає 0, блокуючи хибну генерацію тривожних сповіщень.

На основі отриманого значення визначається рівень прихильності пацієнта:

Рівень	Значення
Високий	90–100 %
Середній	70–89 %
Низький	менше 70 %

У разі падіння загального рівня комплаєнсу пацієнта нижче критичної межі у 70%, програмний модуль `maybeCreateLowComplianceAlert` ініціює автоматичне створення системного попередження для лікуючого лікаря в таблиці `alerts` із кодом правила `low_compliance`. Модуль містить захист від дублювання та генерує нове попередження лише за умови відсутності аналогічних записів за останні 24 години.

Отримані метрики та розраховані часові тренди використовуються для побудови інтерактивних аналітичних панелей пацієнта та лікаря на основі бібліотеки Recharts (`LineChart`, `BarChart`, `PieChart`), відображаючи динаміку показників та тижневу прихильність терапії.

2.2.4 Підсистема клінічних сповіщень, документів та контролю доступу

У сучасних системах дистанційного медичного супроводу важливим завданням є автоматичне виявлення потенційно небезпечних змін стану пацієнта. Для реалізації даного механізму у системі «MedAlert» розроблено підсистему клінічного тріажу, яка аналізує результати вимірювань, що надходять від пацієнта під час виконання реабілітаційних завдань.

Для автоматичного моніторингу критичних ситуацій реалізовано таблицю `alerts`.

```
alerts (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
patient_id UUID NOT NULL REFERENCES public.profiles(id) ON DELETE CASCADE,
doctor_id UUID NOT NULL REFERENCES public.profiles(id) ON DELETE CASCADE,
source_log_id UUID REFERENCES public.task_logs(id) ON DELETE SET NULL,
category TEXT NOT NULL, -- 'warning' або 'critical'
title TEXT NOT NULL,
status TEXT NOT NULL DEFAULT 'new', -- 'new', 'in_progress', 'resolved'
source_rule TEXT,       -- 'triage' або 'low_compliance'
details JSONB NOT NULL DEFAULT '{}'::jsonb,
comment TEXT,
resolved_at TIMESTAMPTZ,
resolved_by UUID REFERENCES public.profiles(id),
created_at TIMESTAMPTZ NOT NULL DEFAULT now()
);
```

Підтримуються категорії повідомлень:

```
Critical – критичне;
Warning – попереджувальне;
System – системне.
```

Стан обробки повідомлення:

```
New –                нове;
in_progress –        взято в роботу;
resolved –           вирішено;
escalated –          ескальовано;
closed_no_response – закрито без зв'язку.
```

Система автоматично формує сповіщення при виникненні подій, що потребують уваги лікаря, зокрема:

- пропуск завдань пацієнтом;
- зниження рівня комплаєнсу;
- виявлення критичних медичних показників;

- службові системні повідомлення.

Для оцінки стану пацієнта використовується механізм класифікації ризику за трьома рівнями:

Green — показники знаходяться в межах норми;

Yellow — зафіксовано помірні відхилення;

Red — виявлено критичні значення показників.

Під час аналізу оцінюються такі параметри:

- артеріальний тиск;
- частота серцевих скорочень;
- сатурація кисню крові;
- температура тіла;
- інтенсивність больового синдрому.

Алгоритм тріажу (src/lib/triage.ts) парсить транзакційний payload на відповідність медичним кордонам ризику:

- Систолічний АТ (bp_sys): > 180 мм рт. ст. (red), > 140 мм рт. ст. (yellow);
- Діастолічний АТ (bp_dia): > 120 мм рт. ст. (red), > 90 мм рт. ст. (yellow);
- Сатурація (SpO_2): < 90% (red), < 94% (yellow);
- Частота пульсу: < 40 або > 130 уд/хв (red), < 50 або > 110 уд/хв (yellow);
- Рівень болю: ≥ 9 балів (red), ≥7 балів (yellow).

Проєктне рішення підсистеми сповіщень враховує ризики інформаційного перевантаження лікаря (Alert Fatigue). На рівні інтеграційної логіки реалізовано механізм дедуплікації інцидентів. При фіксації тригерного стану система виконує превентивний аналіз таблиці `public.alerts`. Нове сповіщення генерується виключно у разі відсутності аналогічного відкритого інциденту за тією ж категорією та метрикою для конкретного пацієнта в межах встановленого часового вікна (12 годин для клінічного тріажу показників та 24 години для аналітики комплаєнсу). Для контролю всіх операцій та змін у картах пацієнтів використовується таблиця `audit_logs`, що забезпечує повне журналювання дій користувачів.

2.2.5 Захист медичних даних та контроль доступу

Для зберігання клінічних сповіщень використовується таблиця `alerts`. Кожне сповіщення містить:

- категорію повідомлення;
- заголовок;
- детальний опис;
- ідентифікатор лікаря;
- ідентифікатор пацієнта;
- статус опрацювання;
- посилання на джерело виникнення події.

Для забезпечення простежуваності клінічних подій використовується поле `source_log_id`, яке пов'язує сповіщення із відповідним записом журналу виконання.

Медичні документи пацієнтів зберігаються у таблиці `patient_documents`, що дозволяє прикріплювати результати аналізів, висновки обстежень та інші файли.

Контроль усіх важливих дій у системі здійснюється через таблицю `audit_logs`, у якій реєструються:

- створення та редагування планів;
- внесення медичних показників;
- створення сповіщень;
- інші критично важливі операції користувачів.

```
audit_logs (
  id UUID PRIMARY KEY,
  actor_id UUID,
  patient_id UUID,
  entity TEXT,
  action TEXT,
  details JSONB,
  created_at TIMESTAMPTZ
)
```

Наявність журналу аудиту забезпечує можливість відстеження історії змін та підвищує надійність функціонування інформаційної системи.

Захист медичних даних

Особливу увагу під час проєктування бази даних приділено безпеці медичної інформації. Розмежування прав доступу реалізовано за допомогою механізму Row Level Security (RLS) у PostgreSQL.

У межах прототипу лікар має доступ до базового реєстру пацієнтів для формування індивідуального плану реабілітації. Водночас доступ до клінічних записів, журналів виконання, документів і сповіщень обмежується відповідними зв’язками між лікарем, пацієнтом і активним планом лікування.

Узагальнено політики доступу реалізують такі правила:

1. Пацієнт має доступ до власного профілю, медичних документів, журналів виконання та персональних сповіщень.
2. Лікар має доступ до пацієнтів і клінічних даних у межах ведення планів реабілітації.
3. Обробка клінічних сповіщень дозволена лише авторизованому лікарю.
4. Файли зберігаються у приватних сховищах Supabase Storage і не відкриваються як публічні ресурси.
5. Службові та клінічні дії можуть фіксуватися у журналі аудиту.

Файли зберігаються у приватному сховищі `patient_documents`, доступ до якого контролюється політиками безпеки Supabase.

```
patient_documents (
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
patient_id UUID NOT NULL REFERENCES public.profiles(id) ON DELETE CASCADE,
name TEXT NOT NULL,
doc_type TEXT,
doc_date DATE,
comment TEXT,
file_path TEXT NOT NULL,
created_at TIMESTAMPTZ DEFAULT now()
);
```

Фізичні файли завантажуються у приватний бакет `patient-files` у Supabase Storage. Доступ до документів мають лише авторизовані користувачі відповідно до політик доступу: пацієнт-власник документа та лікар, який працює з відповідним пацієнтом.

Таким чином, спроєктована структура бази даних забезпечує збереження профілів користувачів, медичних анкет, професійних даних лікарів, планів реабілітації, щоденних завдань, журналів виконання, документів, аналітичних подій та клінічних сповіщень. Використання реляційних зв’язків у поєднанні з JSONB-полями дозволяє підтримувати як структуровані медичні дані, так і гнучкі конфігурації індивідуальних реабілітаційних програм.

2.3 Реалізація програмної архітектури системи

Архітектура програмної системи «MedAlert» реалізована за клієнт-серверною моделлю з використанням сучасного стеку вебтехнологій. Клієнтська частина побудована на основі бібліотеки React із використанням мови TypeScript, що забезпечує компонентний підхід до розробки інтерфейсу та статичну типізацію даних. Основу проєкту становить середовище Vite, яке використовується для швидкої збірки та запуску застосунку під час розробки. Як серверну платформу обрано Supabase (Backend-as-a-Service), який забезпечує хмарне зберігання даних, механізми автентифікації користувачів, обробку сховищ файлів та взаємодію з реляційною базою даних PostgreSQL.

Архітектурно система складається з чотирьох основних рівнів:

1. *Рівень представлення (Presentation Layer)* – реалізує взаємодію користувача із системою за допомогою React-компонентів. Для побудови інтерфейсу використано бібліотеку React, фреймворк Tailwind CSS та набір UI-компонентів Shadcn/UI.
2. *Рівень бізнес-логіки (Business Logic Layer)* – містить програмні модулі обробки даних, користувацькі React-хуки, сервіси аналізу медичних показників, механізми розрахунку комплаєнсу пацієнтів та алгоритми автоматизованого клінічного триажу.
3. *Рівень доступу до даних (Data Access Layer)* – забезпечує взаємодію клієнтської частини із серверною інфраструктурою через Supabase JavaScript API.

Для керування асинхронними запитами та кешуванням даних використовується бібліотека @tanstack/react-query.

4. *Рівень збереження даних (Database Layer)* – представлений реляційною базою даних PostgreSQL, у якій реалізовано механізми цілісності даних, зовнішні ключі, індекси та політики безпеки Row Level Security (RLS), що забезпечують розмежування доступу між ролями лікаря та пацієнта.

У межах програмної реалізації було застосовано компонентну архітектуру React, у якій кожен функціональний екран представлений окремим компонентом сторінки. Вхідною точкою клієнтського застосунку є файл `main.tsx`, де виконується монтування кореневого компонента `App` у DOM-елемент із ідентифікатором `root`:

```
import { createRoot } from "react-dom/client";
import App from "./App.tsx";
import "./index.css";
createRoot(document.getElementById("root")!).render(<App />);
```

Цей фрагмент коду демонструє стандартний механізм ініціалізації React-застосунку: компонент `App` виступає головним контейнером усієї системи, а файл `index.css` підключає глобальні стилі інтерфейсу.

Окрім запуску основного застосунку, у файлі `main.tsx` реалізовано базову підтримку Progressive Web App. Для цього після завантаження сторінки виконується реєстрація service worker, що дозволяє браузеру розпізнавати застосунок як PWA та забезпечує основу для подальшої реалізації offline-режиму й системних сповіщень.

```
 if ("serviceWorker" in navigator) {
window.addEventListener("load", () => {
  navigator.serviceWorker.register("/sw.js").catch(() => {});
});
}
```

Таким чином, у системі передбачено можливість PWA-реєстрації, але з урахуванням особливостей тестового середовища розробки.

Основна конфігурація застосунку зосереджена у файлі `App.tsx`. Саме тут підключаються глобальні провайдери, маршрутизація, контекст авторизації, локалізація та компоненти системних повідомлень:

```
    const queryClient = new QueryClient();
const App = () => (
  <QueryClientProvider client={queryClient}>
    <I18nProvider>
      <TooltipProvider>
        <Toaster />
        <Sonner />
        <BrowserRouter>
          <AuthProvider>
            <Routes>
              ...
            </Routes>
          </AuthProvider>
        </BrowserRouter>
      </TooltipProvider>
    </I18nProvider>
  </QueryClientProvider>
);
```

У цій структурі `QueryClientProvider` відповідає за роботу з асинхронними запитами та кешуванням даних, `I18nProvider` забезпечує підтримку локалізації інтерфейсу, `TooltipProvider` використовується для коректної роботи підказок, а `AuthProvider` надає доступ до інформації про авторизованого користувача в усіх дочірніх компонентах.

Маршрутизація реалізована за допомогою компонента `Routes`. Вона розділяє інтерфейс системи на загальні сторінки, сторінки пацієнта та сторінки лікаря:

```
 <Routes>
<Route path="/" element={<Index />} />
<Route path="/auth" element={<Auth />} />
<Route path="/role" element={<RoleSelect />} />

<Route path="/onboarding/patient" element={<PatientOnboarding />} />
<Route path="/onboarding/doctor" element={<DoctorOnboarding />} />

<Route path="/patient" element={<PatientHome />} />
<Route path="/patient/plan" element={<PatientToday />} />
<Route path="/patient/progress" element={<PatientProgress />} />
<Route path="/patient/documents" element={<PatientDocuments />} />
<Route path="/patient/doctor" element={<PatientDoctor />} />
<Route path="/patient/profile" element={<PatientProfile />} />
<Route path="/patient/history" element={<PatientHistory />} />

<Route path="/doctor" element={<DoctorHome />} />
<Route path="/doctor/patients" element={<DoctorPatients />} />
<Route path="/doctor/patients/:id" element={<PatientDetail />} />
<Route path="/doctor/plans" element={<DoctorPlans />} />
<Route path="/doctor/plans/new" element={<PlanNew />} />
<Route path="/doctor/plans/:id" element={<PlanDetail />} />
<Route path="/doctor/alerts" element={<DoctorAlerts />} />
<Route path="/doctor/calendar" element={<DoctorCalendar />} />
<Route path="/doctor/analytics" element={<DoctorAnalytics />} />
<Route path="/doctor/profile" element={<DoctorProfile />} />

<Route path="*" element={<NotFound />} />
</Routes>
```

Така реалізація дозволяє чітко відокремити робочі сценарії лікаря та пацієнта. Пацієнт має доступ до власного плану реабілітації, прогресу, документів, профілю, історії та інформації про лікаря. Лікар, у свою чергу, працює з реєстром пацієнтів, картками пацієнтів, планами реабілітації, критичними сповіщеннями, календарем, аналітикою та власним професійним профілем.

Для підтримки сумісності старих маршрутів із новою структурою використано перенаправлення через компонент `Navigate`:

```
  <Route
  path="/patient-dashboard"
  element={<Navigate to="/patient" replace />}
/>

<Route
  path="/doctor-dashboard"
  element={<Navigate to="/doctor" replace />}
/>

<Route
  path="/doctor/care-plans"
  element={<Navigate to="/doctor/plans" replace />}
/>
```

Це дозволяє уникнути помилок переходу, якщо в системі залишилися старі посилання або користувач відкриває попередній маршрут вручну.

Інтеграція клієнтської частини з Supabase реалізована у файлі `src/integrations/supabase/client.ts`. Клієнт створюється один раз і надалі імпортується у всі компоненти, які працюють із базою даних або автентифікацією:

```
   import { createClient } from "@supabase/supabase-js";
import type { Database } from "./types";

const SUPABASE_URL = import.meta.env.VITE_SUPABASE_URL;
const SUPABASE_PUBLISHABLE_KEY =
  import.meta.env.VITE_SUPABASE_PUBLISHABLE_KEY;

export const supabase = createClient<Database>(
  SUPABASE_URL,
  SUPABASE_PUBLISHABLE_KEY,
  {
    auth: {
      storage: localStorage,
      persistSession: true,
      autoRefreshToken: true,
    },
  }
);
```

У цьому фрагменті важливими є три параметри автентифікації. Параметр `storage: localStorage` дозволяє зберігати сесію користувача у локальному сховищі браузера. Параметр `persistSession: true` забезпечує збереження

авторизованого стану після оновлення сторінки. Параметр `autoRefreshToken: true` дозволяє автоматично оновлювати токен доступу без повторного входу користувача в систему.

Для централізованого доступу до інформації про поточний сеанс реалізовано власний React-хук `useAuth`, розміщений у файлі `src/hooks/useAuth.tsx`. У ньому створено контекст авторизації:

```
   type AuthCtx = {
  user: User | null;
  session: Session | null;
  loading: boolean;
};

const Ctx = createContext<AuthCtx>({
  user: null,
  session: null,
  loading: true,
});
Компонент AuthProvider підписується на зміну стану автентифікації через Supabase Auth:
useEffect(() => {
  const {
    data: { subscription },
  } = supabase.auth.onAuthStateChange((_event, s) => {
    setSession(s);
    setUser(s?.user ?? null);
  });

  supabase.auth.getSession().then(({ data: { session: s } }) => {
    setSession(s);
    setUser(s?.user ?? null);
    setLoading(false);
  });

  return () => subscription.unsubscribe();
}, []);
```

Ця логіка забезпечує два важливі процеси. По-перше, під час відкриття застосунку виконується перевірка поточної сесії користувача через `getSession()`. По-друге, система реагує на зміну стану авторизації через `onAuthStateChange()`, тобто при вході або виході користувача всі компоненти автоматично отримують актуальні дані про поточного користувача.

Доступ до контексту здійснюється через хук:

```
export const useAuth = () => useContext(Ctx);
```

Завдяки цьому будь-який компонент системи може отримати інформацію про користувача без дублювання логіки авторизації. Наприклад, у модулі створення плану реабілітації лікар отримує власний ідентифікатор через:

```
const { user } = useAuth();
```

Після цього `user.id` використовується як `doctor_id` під час створення запису в таблиці `care_plans`.

Важливою частиною програмної архітектури є рольове розділення інтерфейсу. Воно реалізовано не лише через маршрути, але й через різні оболонки сторінок: для лікаря використовується `DoctorShell`, а для пацієнта — `PatientShell`. Такий підхід дозволяє мати окремі бокові меню, набір вкладок і логіку навігації для кожної категорії користувачів.

Структура лікарського кабінету включає такі основні екрани:

`src/pages/doctor/`

- `DoctorHome.tsx` - головна робоча панель лікаря з поточними статистичними показниками;
- `Patients.tsx` - модуль менеджменту та фільтрації закріплених пацієнтів;
- `PatientDetail.tsx` - детальна медична карта пацієнта з історією спостережень;
- `Plans.tsx` та `PlanNew.tsx` - інтерфейси перегляду та конструювання нових програм реабілітації;
- `PlanDetail.tsx` - модуль детального аудиту, коригування та версіонування діючих планів лікування;
- `Alerts.tsx` - екран моніторингу та обробки автоматично згенерованих клінічних сповіщень;
- `Calendar.tsx` – календар подій лікаря;
- `Profile.tsx` – профіль лікаря.

Для пацієнта передбачено окремий набір сторінок, розміщених у каталозі `src/pages/patient/`, що містить:

- `Home.tsx` - головна сторінка пацієнта та щоденний план завдань;
- `Today.tsx` - хронологічна сітка на поточну добу;
- `Progress.tsx` - графіки моніторингу життєво важливих показників;
- `History.tsx` – історія активності;
- `Documents.tsx` - захищене сховище медичних виписок;

– `Doctor.tsx` - інформація про лікаря, який веде план реабілітації;
– `Profile.tsx` - антропометрична та демографічна анкета пацієнта.

Такий розподіл відповідає функціонально-рольовій моделі системи: лікар керує планами, переглядає пацієнтів та аналітику, тоді як пацієнт виконує призначені завдання, переглядає прогрес і завантажує медичні документи.

Окрему роль у системі відіграє каталог `src/lib`, у якому розміщено незалежні сервіси бізнес-логіки. До них належать модулі роботи з планами реабілітації, розрахунку комплаєнсу, клінічного тріажу, перевірки графіка виконання завдань та інших допоміжних операцій. Винесення таких функцій за межі компонентів інтерфейсу дозволяє уникнути надмірного ускладнення React-компонентів і забезпечує повторне використання логіки в різних частинах застосунку.

Прикладом такого підходу є використання окремого сервісу перевірки актуальності завдання:

```
import { isDueToday } from "@/lib/taskSchedule";
```

Цей сервіс використовується на стороні пацієнтського інтерфейсу для визначення, чи потрібно відображати конкретне завдання в поточний день.

Аналогічно, медичний тріаж винесено в окремий модуль:

```
import { runTriage } from "@/lib/triage";
```

а розрахунок комплаєнсу реалізовано через окремий сервіс:

```
import { calcCompliance } from "@/lib/compliance";
```

Тобто React-компоненти відповідають переважно за відображення даних і взаємодію з користувачем, а складні обчислення та перевірки виконуються у незалежних модулях.

Загальна логіка обміну даними в системі має такий вигляд:

1. користувач виконує дію в інтерфейсі;
2. React-компонент отримує дані форми або стану;
3. за потреби викликається допоміжний сервіс із src/lib;
4. сформований об'єкт передається до Supabase через клієнт supabase;

5. результат операції відображається через toast-повідомлення або оновлення стану компонента.

Наприклад, у компоненті створення плану після перевірки обов'язкових полів виконується insert-запит до таблиці `care_plans`, а потім окремим запитом додаються дочірні записи до `care_plan_tasks`. Такий підхід дозволяє чітко розділити головну сутність плану та набір його завдань.

Отже, програмна архітектура системи «MedAlert» побудована за принципами модульності, рольового розмежування та централізованої роботи з даними. Вхідна точка застосунку відповідає за ініціалізацію React і PWA-механізму, App.tsx формує маршрути та глобальні провайдери, useAuth.tsx забезпечує контроль авторизованого користувача, а `client.ts` реалізує єдину точку підключення до Supabase. Така структура дозволяє надалі розширювати систему новими функціональними модулями без порушення вже реалізованої архітектури.

2.4 Програмна реалізація пацієнтського інтерфейсу та підсистеми контролю виконання завдань

Основний функціонал системи «MedAlert» спрямований на автоматизацію взаємодії між лікарем та пацієнтом у процесі реабілітації, моніторингу стану здоров’я та контролю виконання медичних призначень. Ключовим елементом взаємодії з боку користувача є пацієнтський інтерфейс (`src/pages/patient/Home.tsx`), який відображає список призначених на поточну добу завдань і надає механізми їх фіксації. В архітектурі системи дана сторінка виступає центральною точкою збору медичних даних та формування подальших аналітичних процесів.

На початку компонента підключаються основні програмні сервіси системи:

```
import { useAuth } from "@/hooks/useAuth";
import { supabase } from "@/integrations/supabase/client";
import { runTriage, PATIENT_FEEDBACK } from "@/lib/triage";
import { isDueToday } from "@/lib/taskSchedule";
import { moduleById, VITAL_MEASURES } from "@/lib/carePlanModules";
```

Сервіс `useAuth()` забезпечує отримання інформації про авторизованого користувача, `supabase` використовується для взаємодії із серверною частиною, а модуль `runTriage()` виконує автоматичний аналіз фізіологічних показників після їх внесення пацієнтом.

Завантаження та фільтрація активних завдань

Після авторизації система отримує всі активні плани лікування пацієнта та пов'язані із ними завдання з таблиці `care_plan_tasks`. Кожне завдання, що надходить з таблиці, фільтрується за алгоритмом `isDueToday`:

```
import { isDueToday } from "@/lib/taskSchedule"
```

Алгоритм аналізує параметри schedule, що зберігаються у форматі JSONB, та перевіряє відповідність поточному дню тижня, часовому інтервалу та періодичності виконання.

Лише завдання, для яких функція isDueToday() повертає значення true, відображаються в інтерфейсі пацієнта. Кожна дія пацієнта (успішний прийом ліків, введення фізіологічних параметрів) негайно реєструється у транзакційному журналі виконання `task_logs`.

Реалізація механізму відкладення завдань та автоматичного нагадування

Для оптимізації взаємодії пацієнта з інтерфейсом та запобігання психологічному перевантаженню у компоненті реалізовано алгоритм відкладання завдань (`Snooze`).

Стан відкладених елементів зберігається локально у браузері:

```
    const SNOOZE_KEY = "patient_snoozed_tasks";
const getSnoozed = (): Record<string, number> => {
 try {
  return JSON.parse(
   localStorage.getItem(SNOOZE_KEY) || "{}"
  );
 } catch {
  return {};
 }
};
```

Після натискання кнопки відкладення викликається функція:

```
const snooze = (task: any) => {
  setSnoozed(task.id, Date.now() + 30 * 60 * 1000);
  setDelayedIds((s) =>
    new Set(s).add(task.id)
  );
  quietToast(
    "default",
    `Відкладено на 30 хвилин:
     ${task.title}`
  );
  forceTick((x) => x + 1);
};
```

Функція створює часову мітку завершення відкладення та записує її до локального сховища браузера. Це дозволяє зберігати стан навіть після оновлення сторінки.

Після активації режиму `Snooze` система створює два незалежні таймери.

Перший відповідає за предиктивне нагадування:

```
   const t1 = window.setTimeout(() => {
 quietToast(
  "warning",
  `Увага! Через 5 хвилин
   заплановано виконання:
   ${task.title}`
   );
   }, 25 * 60 * 1000);
```

Другий виконує контроль завершення відкладеного періоду:

```
   const t2 = window.setTimeout(async () => {
 const { data: lg } =
 await (supabase as any)
  .from("task_logs")
  .select("status")
  .eq("task_id", task.id)
  .eq("patient_id", user!.id)
  .order("logged_at",
         { ascending: false })
  .limit(1);
  }, 30 * 60 * 1000);
```

Перед створенням запису система перевіряє останній журнал виконання завдання. Якщо статус вже має значення «`done`», додаткові дії не виконуються.

У випадку *відсутності підтвердження виконання* система автоматично створює транзакційний запис у таблиці `task_logs`.

```
    await (supabase as any)
.from("task_logs")
.insert({
 task_id: task.id,
 patient_id: user!.id,
 status: "missed",
 payload: {
  reason: "snooze_expired"
 }
});
```

Таким чином формується повна історія дій пацієнта незалежно від того, чи було завдання виконано успішно.

Реалізація режиму Quiet Hours

Для забезпечення комфортного використання системи у нічний час впроваджено функціональний режим «тиші» (`Quiet Hours`). Спеціальний фільтр аналізує системний час та повністю блокує звукові та візуальні нотифікації (`toast`) у проміжку з 22:00 до 07:00, захищаючи сон та режим відпочинку пацієнта:

```
  const isQuietHours = (d = new Date()) => {
const h = d.getHours();
return h >= 22 || h < 7;
  };
  const quietToast = (kind: "default" | "success" | "error" | "warning", msg: string) => {
if (isQuietHours()) return; // Повне блокування нотифікацій у нічний час
if (kind === "success") toast.success(msg);
else if (kind === "error") toast.error(msg);
else toast(msg);
  };
```

Алгоритм клінічного тріажу показників здоров'я

Інтелектуальне ядро системи «MedAlert» зосереджене у незалежних програмних сервісах `src/lib/triage.ts` та `src/lib/compliance.ts`. Вони виконують потоковий аналіз фізіологічних параметрів, що вносяться пацієнтами, здійснюють математичний розрахунок показників приверженості до лікування (комплаєнсу) та керують підсистемою медичних сповіщень.

Під час збереження щоденного звіту пацієнтом за модулями `vitals` або `pain`, система запускає ізольовану функцію `runTriage`. Вона розбирає об'єкт `payload`, виділяє числення показники та порівнює їх із затвердженими

референтними межами трьох рівнів ризику: `green` (норма), `yellow` (потребує уваги лікаря), `red` (критичний стан, що загрожує життю). Критерії оцінки фізіологічних параметрів представлені у функції `classify`:

```
 function classify(metric: string, raw: any): TriageFinding | null {
const v = typeof raw === "number" ? raw : parseFloat(String(raw ?? "").replace(",", "."));
if (!isFinite(v)) return null;
let level: "yellow" | "red" | null = null;

switch (metric) {
  case "systolic":
    if (v > 180) level = "red"; else if (v > 140) level = "yellow"; break;
  case "diastolic":
    if (v > 120) level = "red"; else if (v > 90) level = "yellow"; break;
  case "spo2":
    if (v < 90) level = "red"; else if (v < 94) level = "yellow"; break;
  case "pulse":
    if (v < 40 || v > 130) level = "red"; else if (v < 50 || v > 110) level = "yellow"; break;
  case "pain":
    if (v >= 9) level = "red"; else if (v >= 7) level = "yellow"; break;
}

if (!level) return null;
return {
  metric,
  metricLabel: LABEL[metric] || metric,
  value: v,
  level,
  recommendation: level === "red" ? "КРИТИЧНО: Негайно зв'яжіться з лікарем або викликайте швидку допомогу!" :
  "Увага: Показник відхилено від норми. Рекомендовано відпочинок.",
};
 }
```

Захист від інформаційного перевантаження лікаря (Дедуплікація алертів)

Для запобігання виникненню ефекту «втоми від сповіщень» (`alert fatigue`) у лікаря, алгоритм `runTriage` містить вбудований фільтр дедуплікації. Перед генерацією нового запису в таблицю `alerts`, система аналізує історію сповіщень за останні 12 годин. Якщо для цього пацієнта за ідентичною метрикою та рівнем ризику вже було створено активне попередження, повторний запис блокується:

```
export async function runTriage(opts: {
  patientId: string; sourceLogId: string; payload: Record<string, any>; module: string;
}): Promise<TriageResult> {
  const result = evaluatePayload(opts.payload);
  if (result.level === "green" || result.findings.length === 0) return result;

  const { data: plans } = await (supabase as any)
    .from("care_plans").select("doctor_id").eq("patient_id", opts.patientId).eq("status", "active").limit(1);
  const doctorId = plans?.[0]?.doctor_id;
  if (!doctorId) return result;

  // Встановлення 12-годинного часового вікна дедуплікації
  const sinceISO = new Date(Date.now() - 12 * 60 * 60 * 1000).toISOString();
  const { data: recent } = await (supabase as any)
    .from("alerts").select("id,details,source_log_id")
    .eq("patient_id", opts.patientId).eq("source_rule", "triage").gte("created_at", sinceISO);

  const inserts: any[] = [];
  for (const f of result.findings) {
    const dup = (recent || []).some((r: any) =>
      r.source_log_id === opts.sourceLogId ||
      (r.details?.metric === f.metric && r.details?.risk_level === f.level)
    );
    if (dup) continue; // Пропуск запису у разі дублювання инциденту

    inserts.push({
      patient_id: opts.patientId,
      doctor_id: doctorId,
      category: f.level === "red" ? "critical" : "warning",
      title: `Відхилення показника: ${f.metricLabel}`,
      status: "new",
      source_log_id: opts.sourceLogId,
      source_rule: "triage",
      details: { risk_level: f.level, metric: f.metric, value: f.value, module: opts.module },
    });
  }

  if (inserts.length) await (supabase as any).from("alerts").insert(inserts);
  return result;
}
```

Розрахунок інтегрального комплаєнсу та аналіз трендів

Для оцінювання загальної ефективності процесу реабілітації система кожні 24 години акумулює історичні дані виконання призначень та вираховує індекс прихильності (комплаєнсу) пацієнта за математичною формулою 2.1, яку розглянули у попередній частині.

Якщо розрахований індекс падає нижче критичної межі у **70%**, спрацьовує проактивний аналітичний тригер `maybeCreateLowComplianceAlert`. Він генерує системне попередження для лікаря про втрату контролю над процесом реабілітации, накладаючи 24-годинне вікно обмеження частоти сповіщень:

```
export async function maybeCreateLowComplianceAlert(patientId: string, score: number): Promise<void> {
  if (score >= 70) return; // Рівень комплаєнсу в межах допустимого

  const { data: plan } = await (supabase as any)
    .from("care_plans").select("doctor_id").eq("patient_id", patientId).eq("status", "active").maybeSingle();
  const doctorId = plan?.doctor_id;
  if (!doctorId) return;

  // Перевірка наявності аналогічного попередження за останні 24 години
  const since = new Date(Date.now() - 24 * 60 * 60 * 1000).toISOString();
  const { data: existing } = await (supabase as any)
    .from("alerts").select("id").eq("patient_id", patientId).eq("source_rule", "low_compliance").gte("created_at", since);

  if (existing && existing.length) return; // Запобігання повторному спаму лікаря

  await (supabase as any).from("alerts").insert({
    patient_id: patientId,
    doctor_id: doctorId,
    category: "warning",
    title: "Критичне зниження прихильності пацієнта до лікування",
    details: { compliance_score: score },
    status: "new",
    source_rule: "low_compliance",
  });
}
```

Результати розрахунків та історичні тренди комплаєнсу інтегруються з екранами аналітики лікаря, що дозволяє йому оперативно коригувати плани лікування, змінювати медикаменти або призначати додаткові візити у разі стійкої негативної тенденції фізіологічних показників.

2.5 Тестування та демонстрація роботи веб-системи «MedAlert»

Для комплексної перевірки працездатності, оцінки інтеграції клієнт-серверної архітектури та валідації реалізованих алгоритмів автоматизації клінічного моніторингу було проведено апробацію вебсистеми цифрового супроводу «MedAlert». Тестування виконувалося шляхом симуляції наскрізного клінічного сценарію реабілітації пацієнта під контролем лікаря. Даний підхід дозволив оцінити функціонування системи не як ізольованих програмних функцій, а як єдиного контуру інформаційного обміну в межах життєвого циклу терапевтичного супроводу.

На першому етапі (рис 2.2) демонструється процес первинного створення облікового запису в системі, оскільки сценарій передбачає взаємодію з абсолютно новими користувачами. Під час реєстрації користувач вводить базові автентифікаційні дані (електронну пошту та пароль) і обов'язково обирає свою майбутню роль у системі: «лікар» або «пацієнт». Таке гнучке розділення на етапі

реєстрації гарантує, що система одразу створює правильний вектор доступу для конкретного суб'єкта медичного процесу.

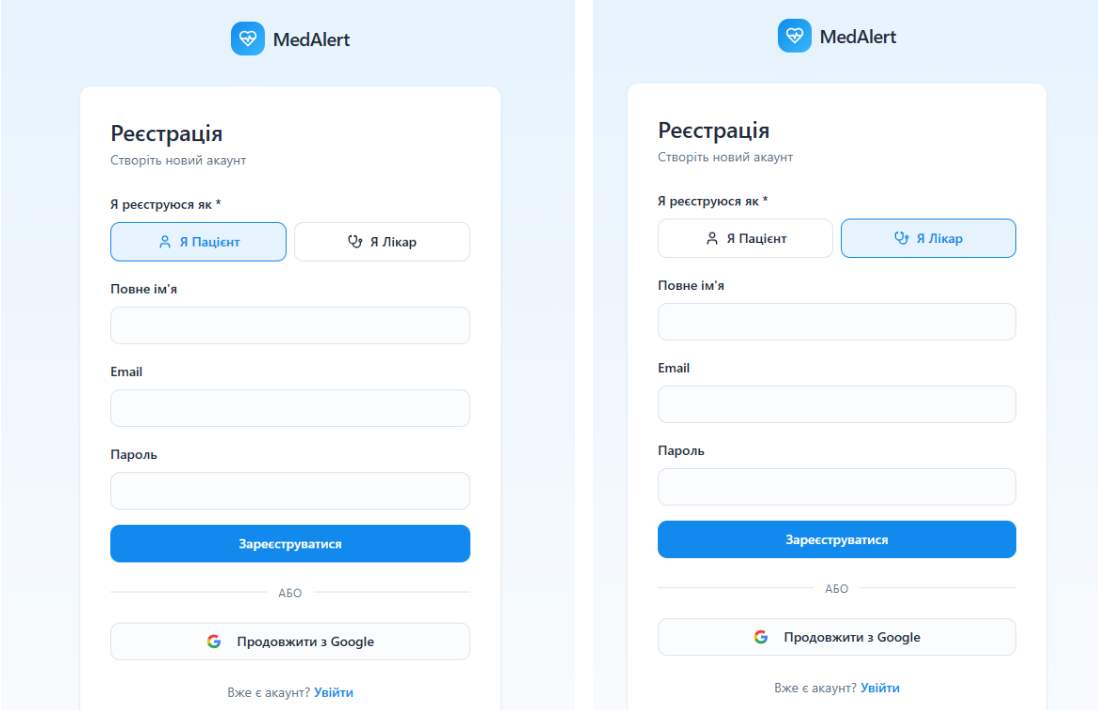


Рисунок 2.2 - Первинне створення облікового запису в системі з вибором ролі

Одразу після успішного підтвердження реєстрації система не допускає користувача до робочих кабінетів, оскільки профіль є пустим. Замість цього запускається обов'язковий крок первинного анкетування користувача. Якщо зареєструвався лікар, йому відкривається інтерфейс заповнення професійного профілю. Якщо ж реєструється пацієнт, система пропонує заповнити базову медичну картку. Тільки після збереження анкетних даних профіль вважається повністю ініціалізованим. Анкети наведені на рисунку 2.3 та 2.4 відповідно.

Після проходження процедури реєстрації та анкетування користувач отримує повноцінний доступ до системи і в подальшому може використовувати стандартну функцію входу. На уніфікованій сторінці авторизації користувач вводить уже наявні в базі даних логін та пароль. Система виконує автоматичну верифікацію особи та її ролі, після чого вебзастосунок здійснює динамічне перенаправлення у відповідний захищений робочий кабінет, що підтверджує коректність функціонування механізмів авторизації та впроваджених правил розмежування доступу.

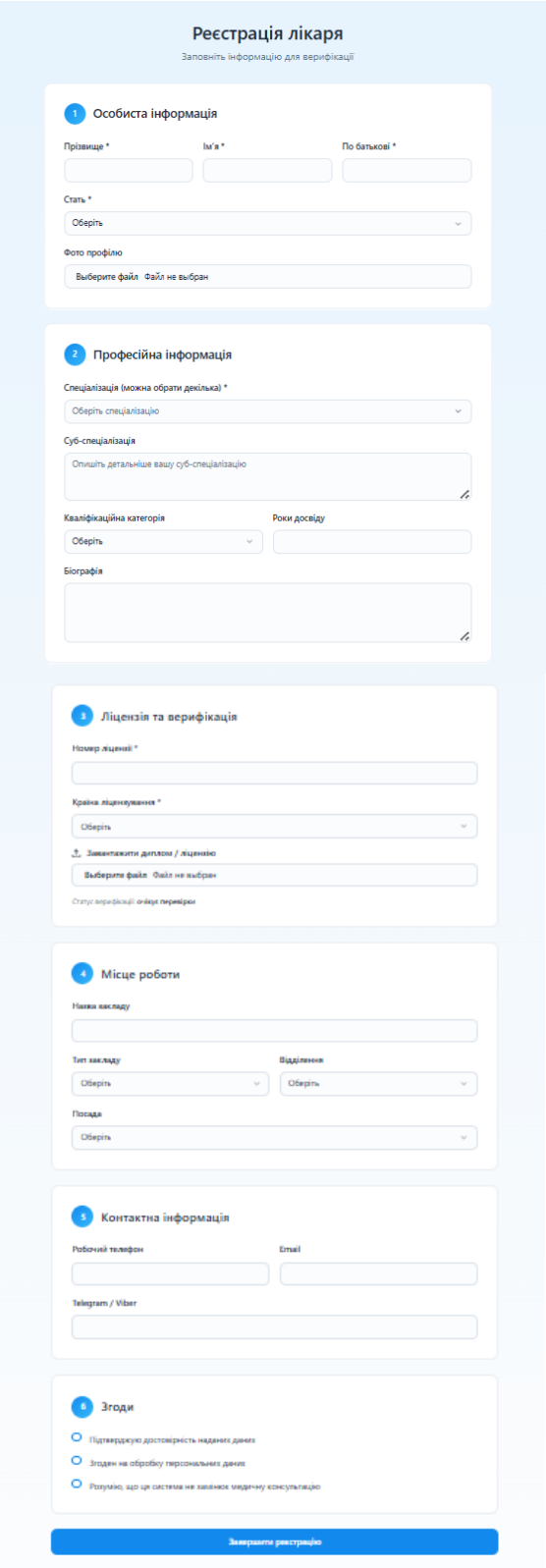


Рисунок 2.3 - Анкета лікаря

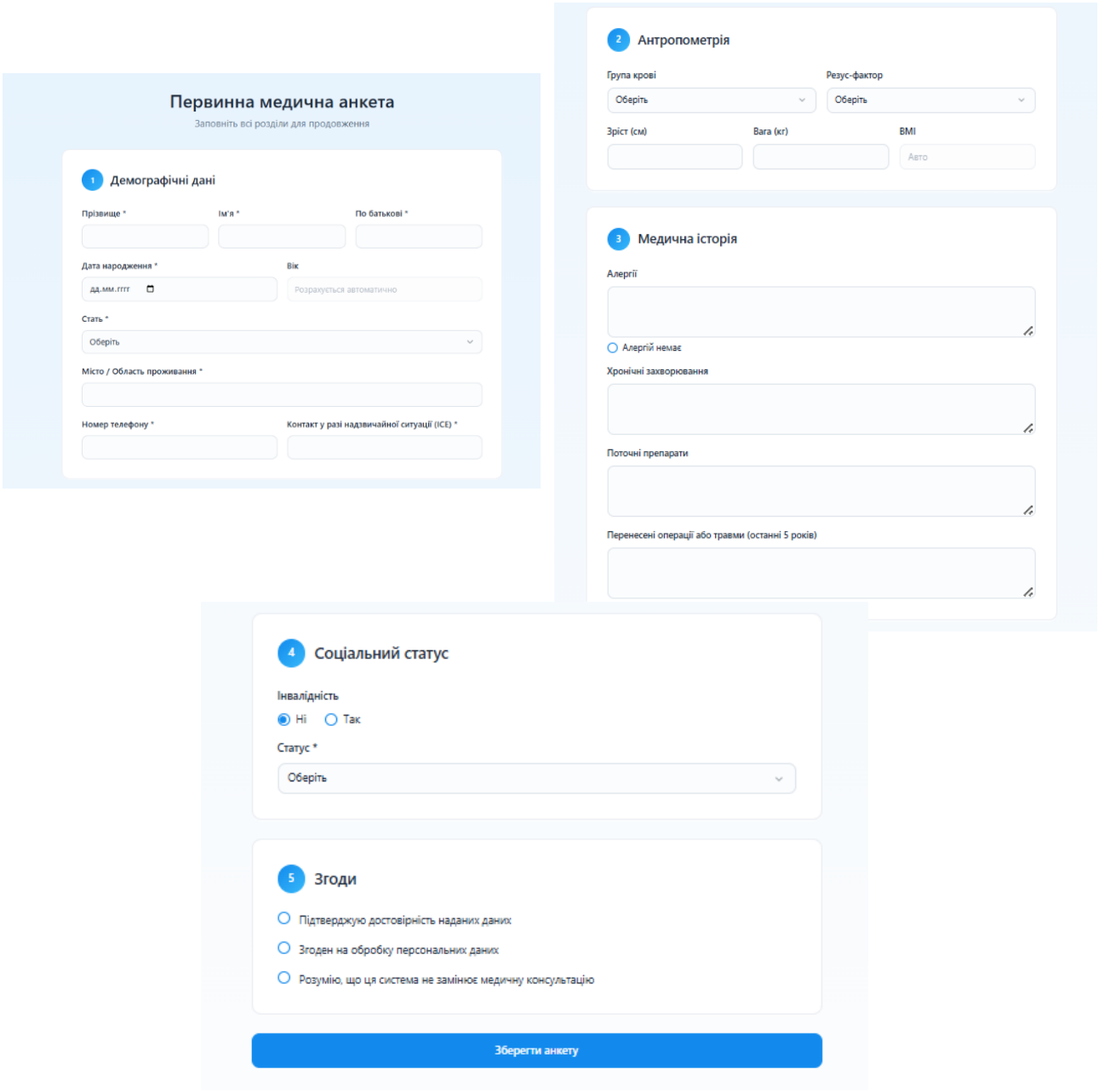


Рисунок 2.4 – Первинна медична анкета

Після авторизації лікаря відкривається головна панель лікарського кабінету. На ній відображаються узагальнені показники: кількість активних пацієнтів, середній рівень виконання призначень, кількість запланованих візитів та критичних повідомлень. Така інформаційна панель дозволяє лікарю швидко оцінити поточний стан роботи із пацієнтами.

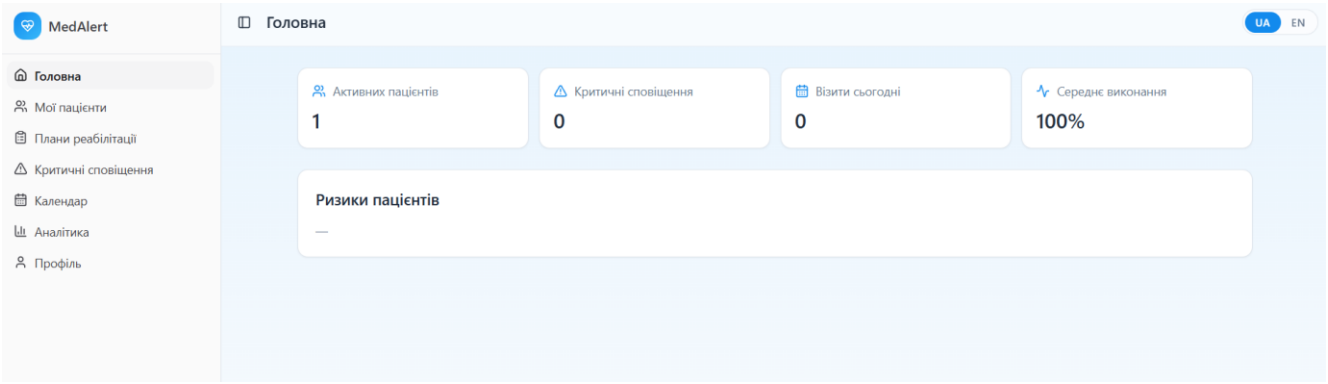


Рисунок 2.5 - Головна сторінка кабінету лікаря

Далі користувач переходить до розділу «Мої пацієнти» (рис 2.6), який складається з двох функціональних блоків: «Нові пацієнти» та «Активні пацієнти». У блоці *нових пацієнтів* відображаються користувачі, які зареєструвалися в системі та доступні для подальшої роботи лікаря. Для кожного такого пацієнта передбачено можливість підтвердження його додавання до робочого списку. Після підтвердження пацієнт автоматично переноситься до розділу активних пацієнтів. У блоці *активних пацієнтів* лікар може переглядати закріплених за ним пацієнтів, відкривати їхні картки для детального перегляду інформації або архівувати завершені випадки спостереження.

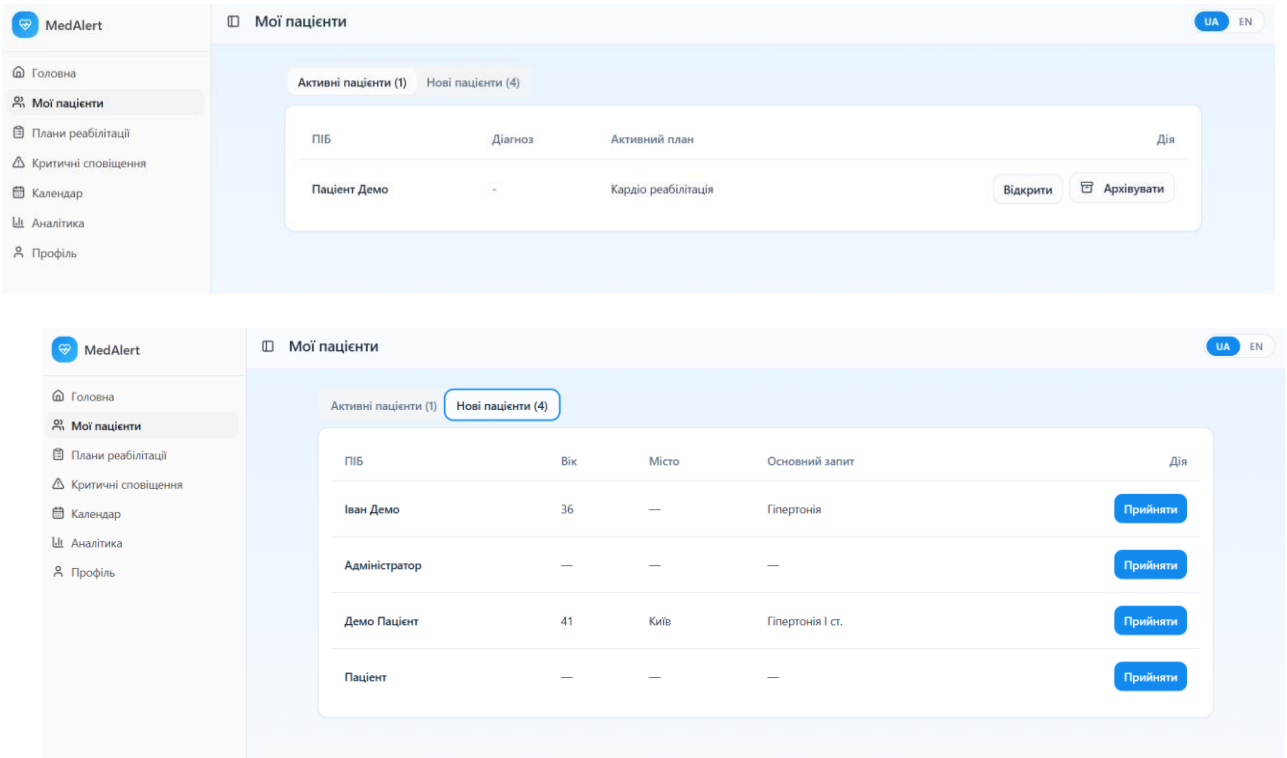
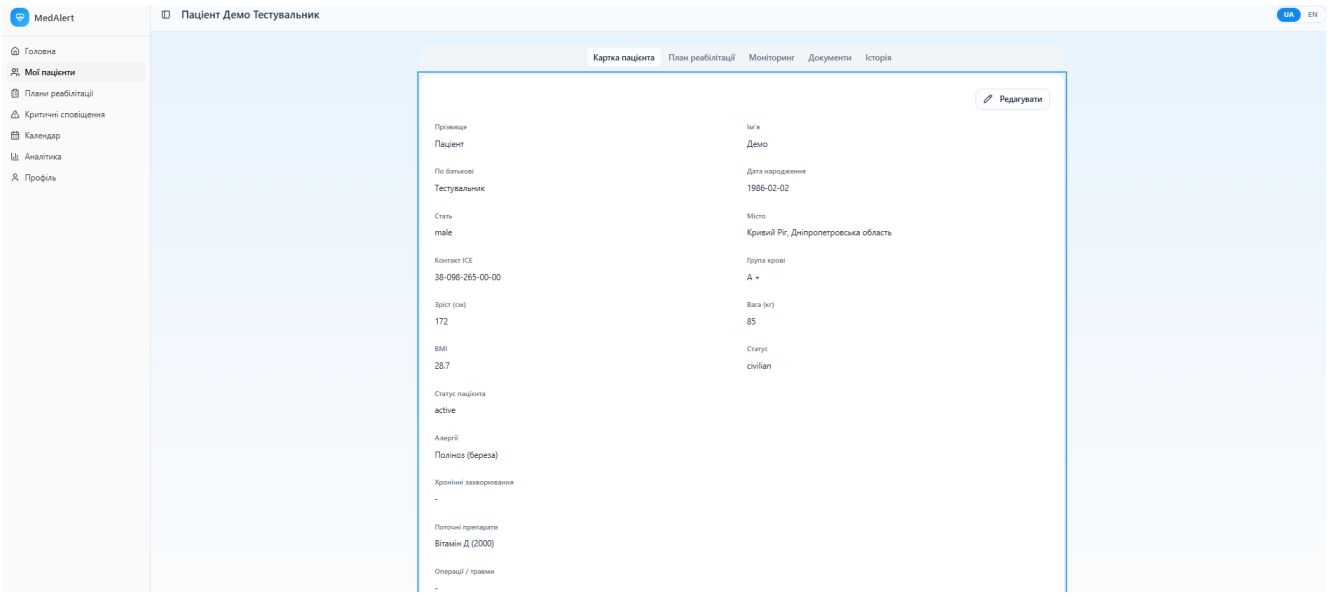
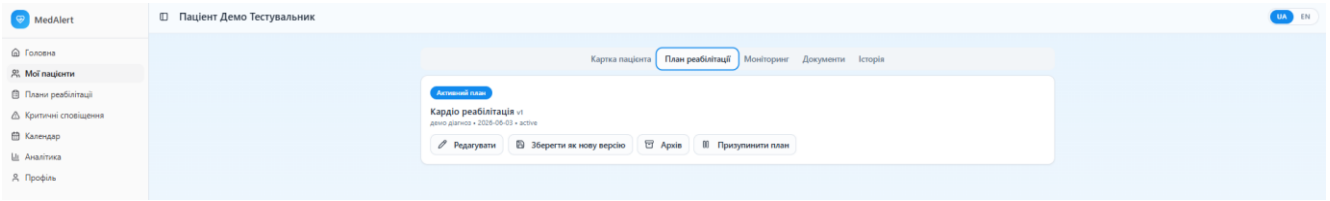


Рисунок 2.6 - Розділ «Мої пацієнти» у кабінеті лікаря

Після відкриття активного пацієнта лікар переходить до його персональної картки (рис 2.7), яка об'єднує основні інструменти спостереження та супроводу реабілітаційного процесу. Інтерфейс картки структуровано у вигляді декількох функціональних вкладок: «Картка пацієнта», «План реабілітації», «Моніторинг», «Документи» та «Історія».

Вкладка «Картка пацієнта» містить основні персональні та медичні дані користувача, необхідні для ведення спостереження. У розділі «План реабілітації» відображається активний план лікування, призначені завдання та доступні інструменти керування планом, зокрема його редагування, призупинення, архівування або створення нової версії. Вкладка «Моніторинг» забезпечує перегляд поточної активності пацієнта та внесених ним медичних показників. У розділі «Документи» лікар може переглядати завантажені пацієнтом медичні файли, результати обстежень та інші супровідні документи. Вкладка «Історія» призначена для перегляду журналу дій та подій, зафіксованих у процесі роботи із системою.





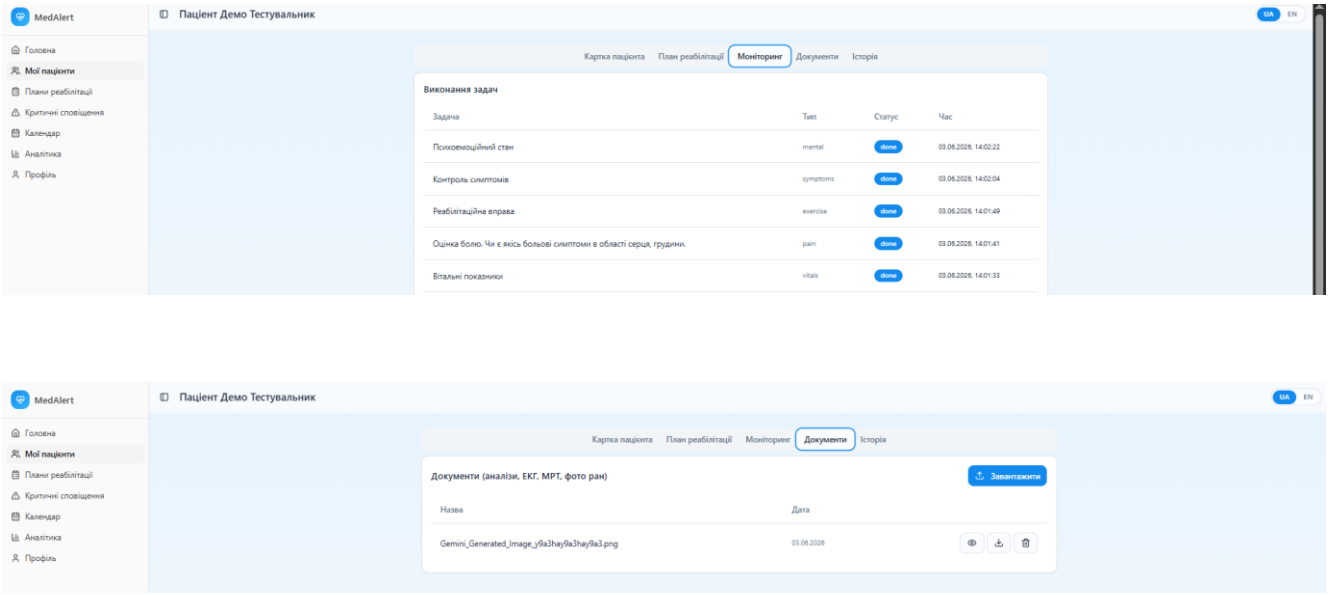


Рисунок 2.7 - Персональна картка активного пацієнта у кабінеті лікаря

Наступним кроком демонструється створення індивідуального плану реабілітації. Лікар обирає пацієнта зі списку, зазначає назву плану, діагноз, дату початку та завершення, після чого додає необхідні модулі лікування й моніторингу.

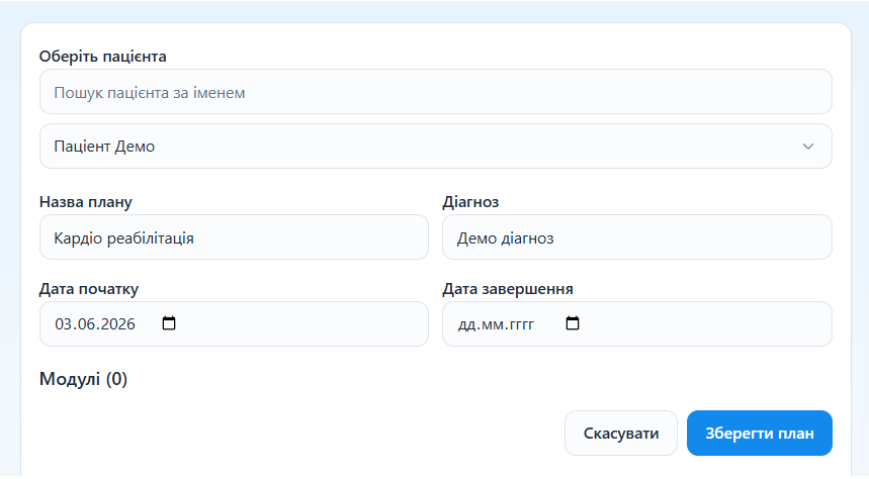


Рисунок 2.8- Форма створення індивідуального плану реабілітації

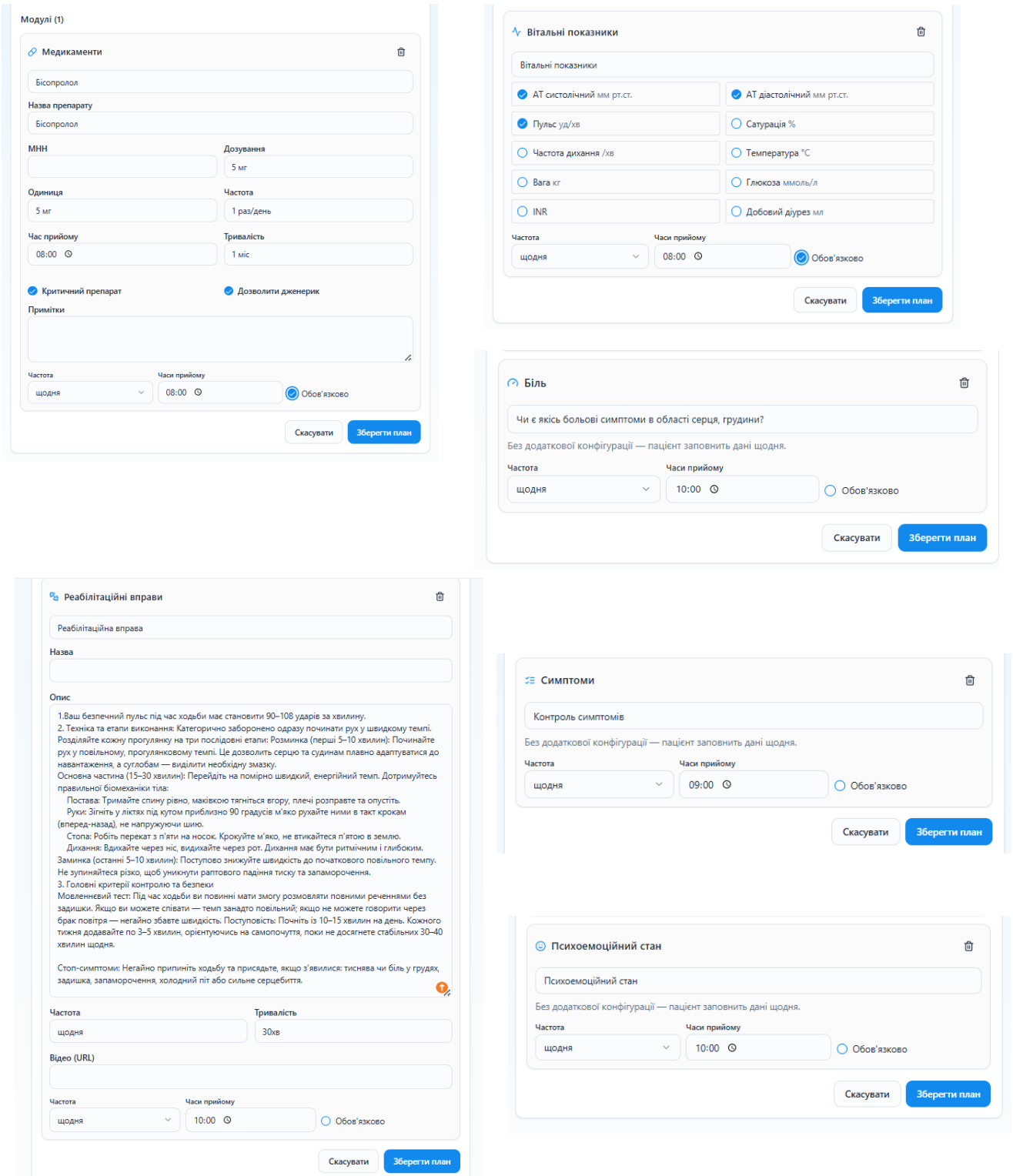


Рисунок 2.9 - Додавання модулів до плану реабілітації

На основі базових параметрів плану лікар здійснює інтерактивне наповнення програми реабілітації шляхом додавання та індивідуального налаштування необхідних модулів лікування й моніторингу (рис 2.9). Архітектура

системи дозволяє гнучко комбінувати різні типи призначень: графік прийому медикаментів, трекер вітальних показників (артеріальний тиск, сатурація, пульс, температура), шкалу оцінки інтенсивності болю, комплекси лікувальних вправ, графік лабораторних аналізів, листи самоконтролю симптомів чи психоемоційного стану (в прикладі наведено додавання 6 модулів). Для кожного модуля лікар задає розклад та ознаку обов'язковості виконання, що виключає жорстке кодування клінічних протоколів.

Після збереження план відображається у списку планів лікаря. Система дозволяє відкрити детальну сторінку плану, переглянути його складові, архівувати або редагувати основні параметри. Таким чином підтверджується зв’язок між лікарем, пацієнтом та створеним планом лікування.

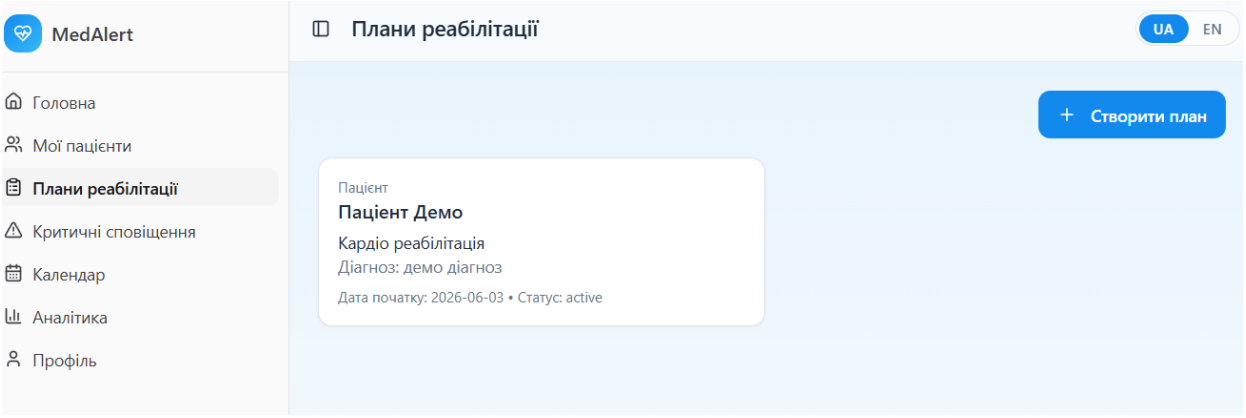


Рисунок 2.10 - Збережений план реабілітації у кабінеті лікаря

Наступна фаза сценарію описує взаємодію із системою з боку пацієнта (після створення лікарем плану виконується вхід до кабінету пацієнта). На головній сторінці пацієнта відображаються завдання, актуальні для поточної доби. До таких завдань належать прийом медикаментів, внесення фізіологічних показників, оцінка болю, виконання вправ або інші активності, сформовані на основі плану лікаря (всі що ми сформували раніше).

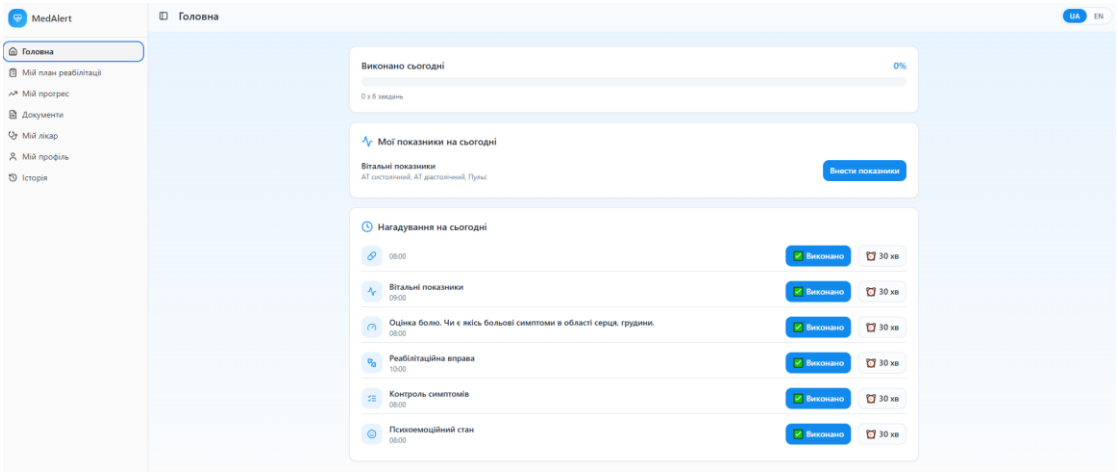
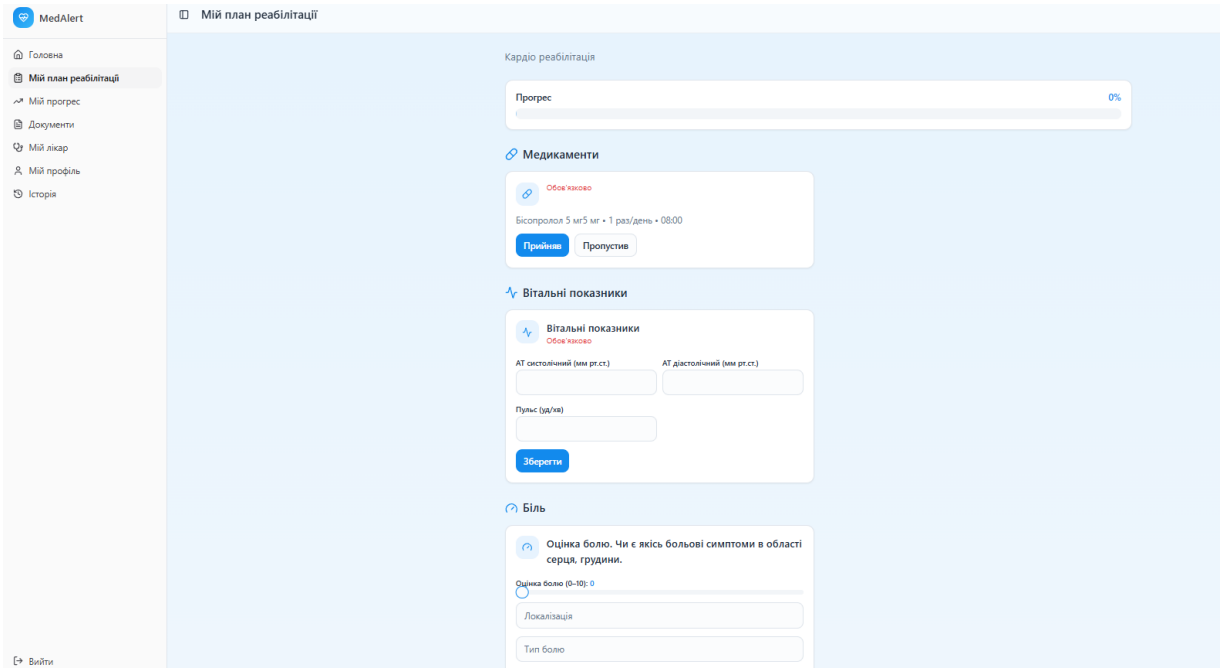


Рисунок 2.11 - Головна сторінка пацієнта із завданнями на день

Для підвищення прозорості лікувального процесу пацієнту доступний окремий розділ «Мій план реабілітації». У цьому модулі відображається вся призначена йому програма відновлення, що дозволяє пацієнту чітко розуміти довготривалу стратегію, загальну тривалість терапії та структуру щоденних завдань.



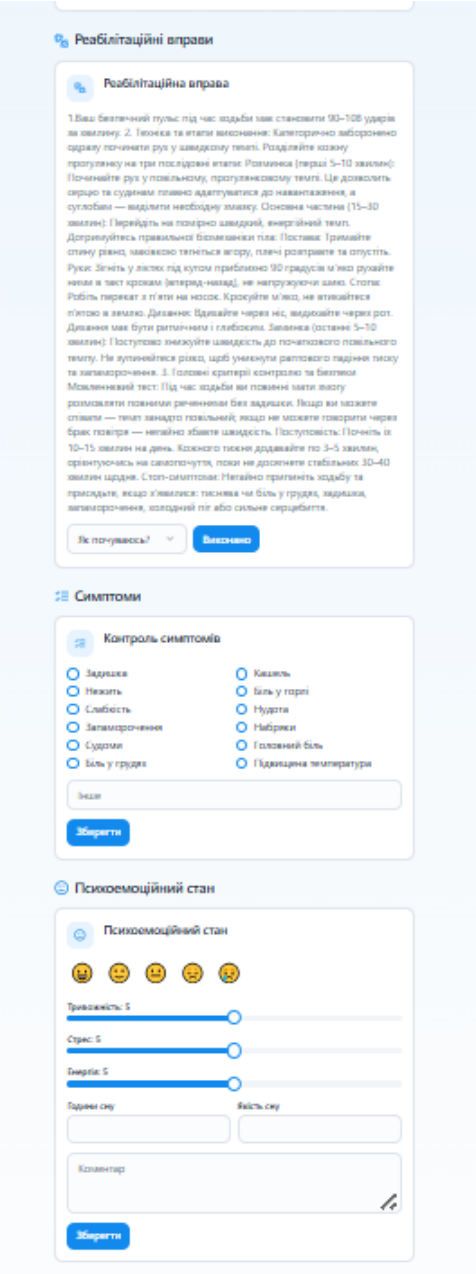


Рисунок 2.12 - План реабілітації у кабінеті пацієнта

Одним із ключових сценаріїв роботи системи є внесення пацієнтом медичних показників. У межах апробації демонструється форма введення значень артеріального тиску, пульсу, сатурації, температури або інших параметрів, які були визначені лікарем у плані реабілітації.

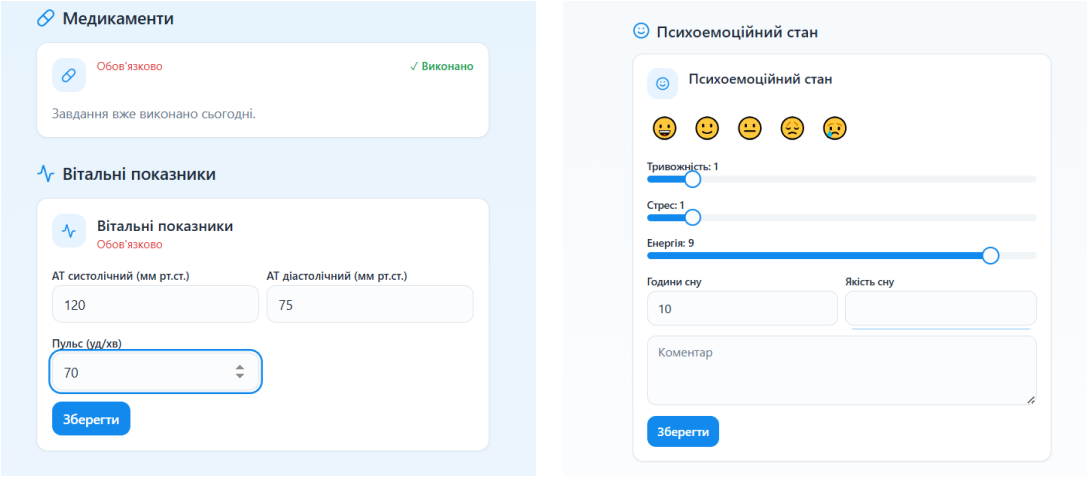


Рисунок 2.13 - Форма внесення медичних показників пацієнтом

Після збереження дані автоматично надходять до загального журналу виконання завдань, а відповідне завдання в інтерфейсі миттєво змінює статус на виконане, маркується зеленим індикатором, а прогрес-бар виконання добового плану автоматично оновлює поточне значення.

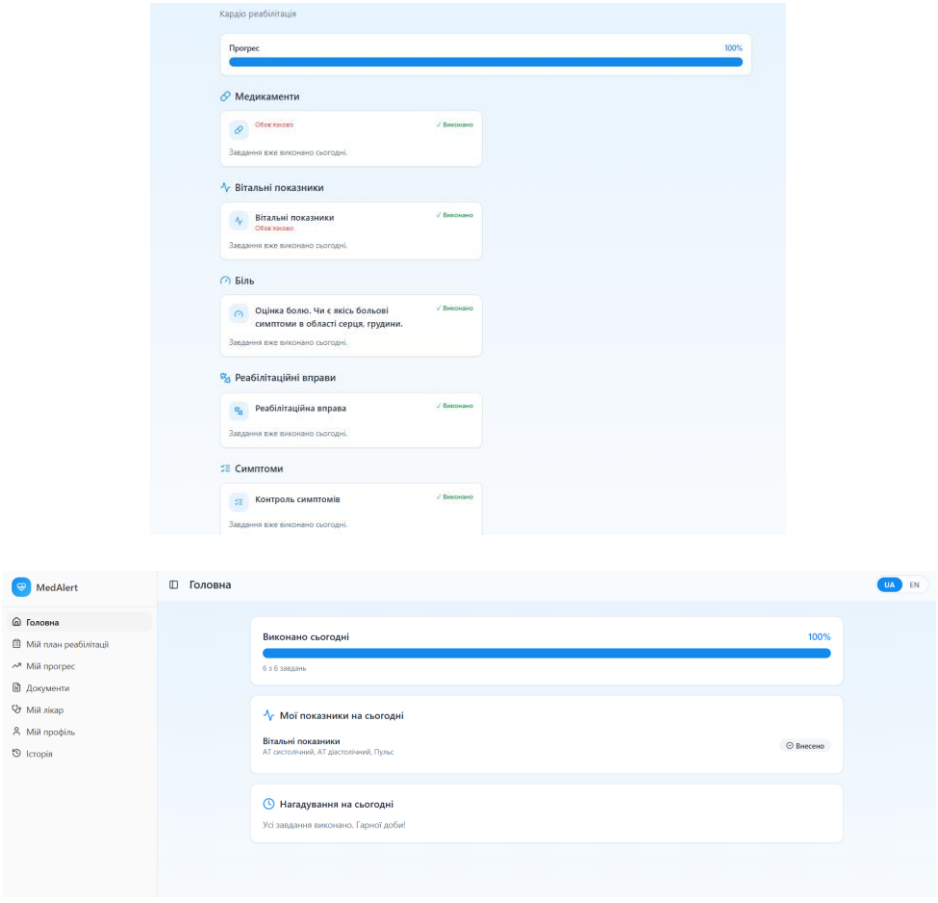


Рисунок 2.14 – Позначення завдання як виконаного

Після виконання завдань пацієнт може переглянути власний прогрес у розділі «Мій прогрес». У цьому розділі відображається відсоток виконання завдань, кількість виконаних і пропущених активностей, тижнева динаміка та інтегральний показник комплаєнсу. Оскільки прототип апробується на обмеженому проміжку часу, графіки відображають демонстраційні дані, сформовані під час проходження сценарію.

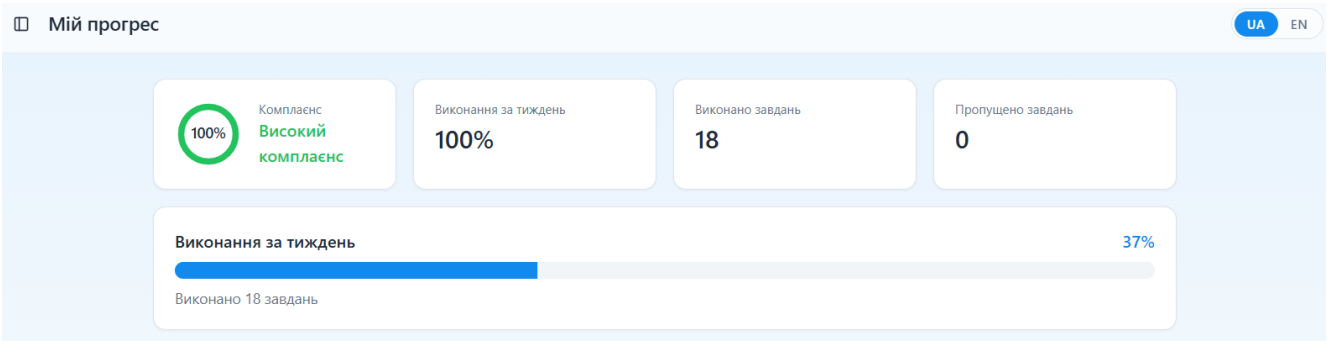


Рисунок 2.15 – Розділ «Мій прогрес» пацієнта

У розділі «Документи» рис 2.16, пацієнт може завантажити медичний документ, вказати його тип, дату та коментар. Такий модуль призначений для зберігання результатів аналізів, виписок, знімків, висновків лікарів та інших файлів, які можуть бути важливими для процесу реабілітації.

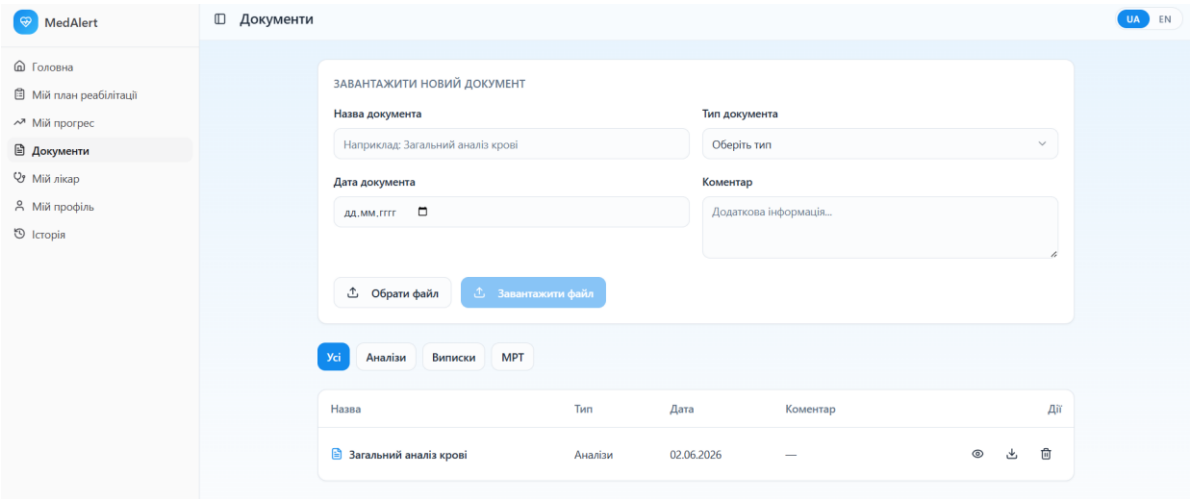


Рисунок 2.16 - Завантаження медичного документа пацієнтом

Також у кабінеті пацієнта передбачено розділ «Мій лікар». У ньому відображається інформація про лікаря, який веде активний план реабілітації: фото, прізвище, ім’я, по батькові, спеціалізація, кваліфікація, місце роботи та контактні дані. Це підтверджує, що зв’язок між пацієнтом і лікарем формується на основі створеного активного плану.

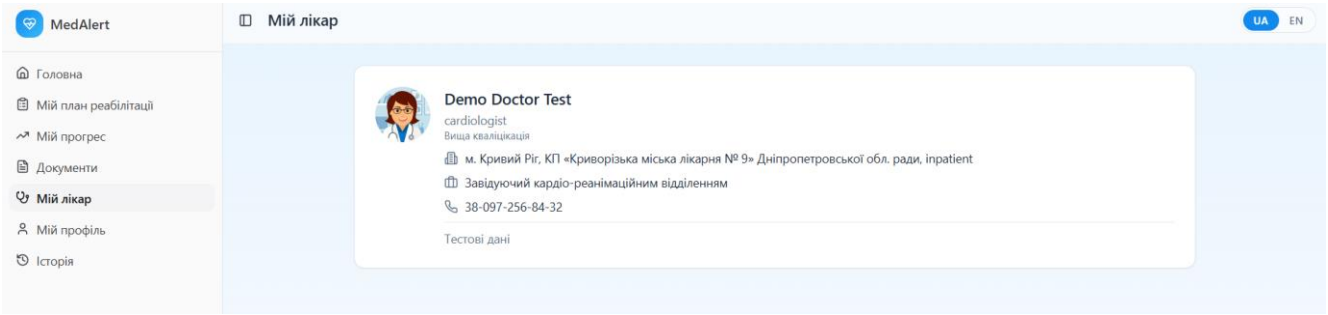


Рисунок 2.17 - Розділ «Мій лікар» у кабінеті пацієнта

Після виконання пацієнтом завдань лікар може переглянути результати у власному кабінеті. У картці пацієнта доступний розділ моніторингу, де відображається історія активності та останні записи пацієнта. Це дозволяє лікарю бачити, які завдання було виконано, коли були внесені показники та які дії здійснював пацієнт у системі.

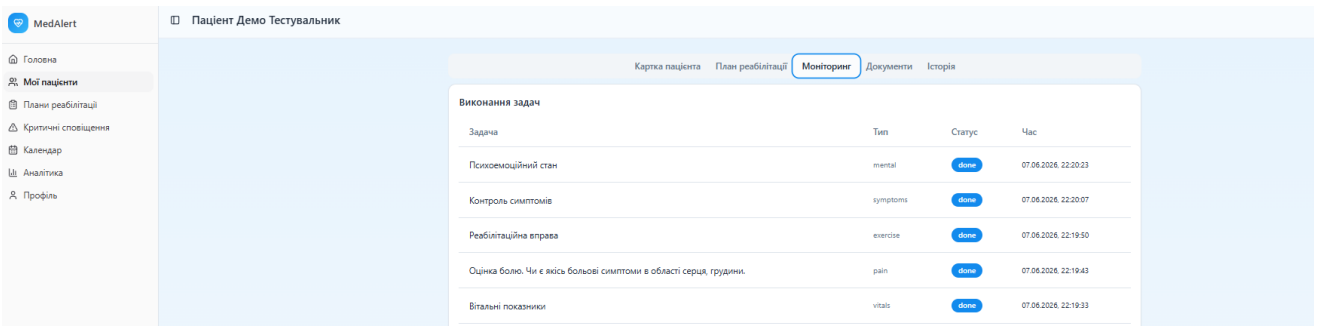


Рисунок 2.18 - Моніторинг активності пацієнта у кабінеті лікаря

Окремо демонструється аналітичний модуль лікаря. У ньому відображаються середній комплаєнс, кількість активних пацієнтів, кількість відкритих сповіщень, графік тижневої прихильності, тренд комплаєнсу та динаміка окремих медичних показників. У межах апробації ці графіки відображають дані, сформовані під час тестового проходження сценарію.

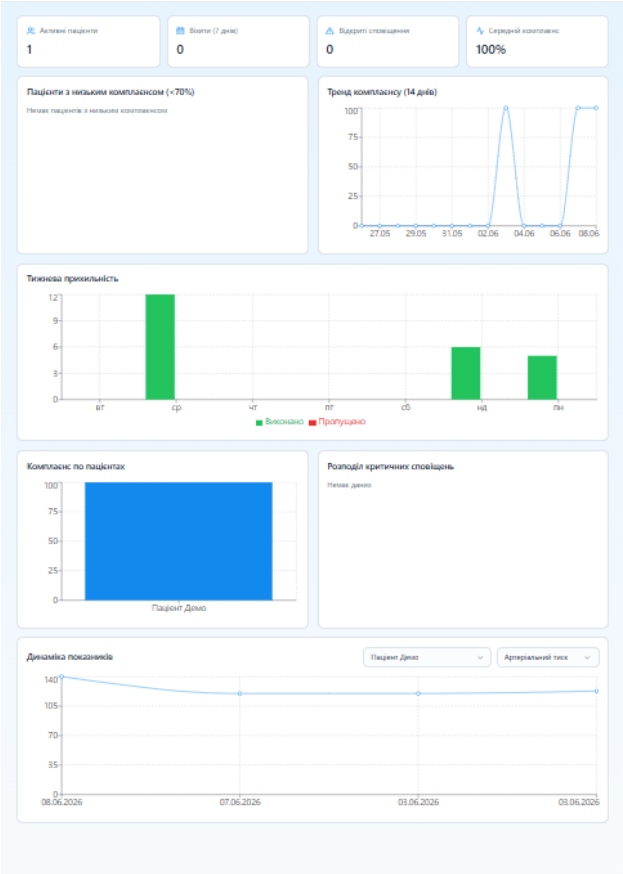


Рисунок 2.19 - Аналітична панель лікаря

У системі також реалізовано підсистему клінічного тріажу та автоматичних сповіщень. Після внесення пацієнтом фізіологічних показників система аналізує їх відповідно до заданих медичних порогових значень. У разі виявлення критичних відхилень від норми система автоматично формує відповідне попередження для лікаря, яке миттєво з'являється в його центрі клінічних сповіщень із зазначенням рівня ризику події.

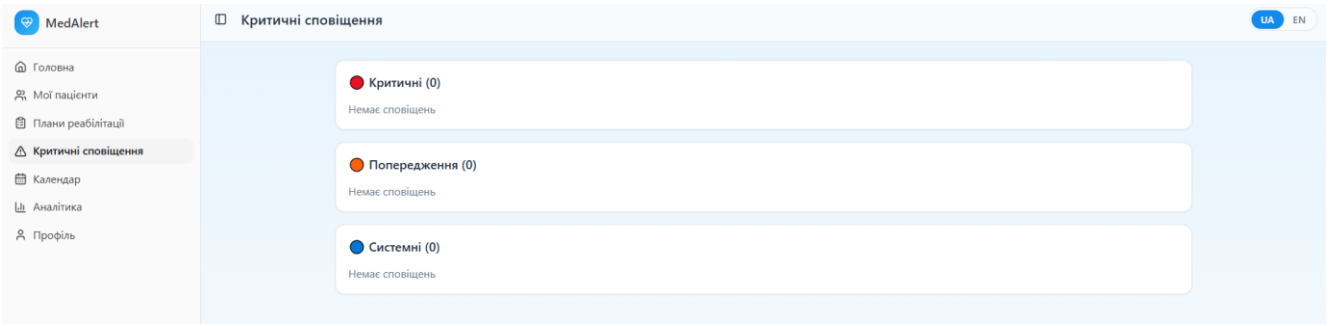


Рисунок 2.20 – Центр клінічних сповіщень лікаря

За потреби лікар може оперативно взаємодіяти зі сповіщенням: взяти його в роботу, ескалувати на вищий рівень або підтвердити вирішення ситуації із додаванням обов'язкового клінічного коментаря щодо вжитих заходів. Це дозволяє відстежувати не лише факт виникнення критичної події, а й процес її подальшої обробки лікуючим фахівцем.

Окрім цього, як лікар, так і пацієнт мають у своїх кабінетах персональний профіль із можливістю редагування особистих чи професійних даних, а також лікар має інтерактивний календар для створення / відстеження подій та повну історію взаємодії в системі.

Проведена апробація показала, що розроблена веб-система «MedAlert» забезпечує проходження основного користувацького сценарію: лікар може створити індивідуальний план реабілітації, пацієнт бачить призначені завдання, виконує їх, вносить медичні показники, переглядає власний прогрес та працює з документами, а лікар отримує доступ до моніторингу й аналітики. Окремі модулі, зокрема довготривале накопичення статистики, клінічний тріаж та автоматичні сповіщення, у межах прототипу демонструють закладену логіку роботи та можуть бути повноцінно апробовані під час подальшого тестування системи на більшій вибірці пацієнтів і тривалішому періоді спостереження.

ВИСНОВКИ

У кваліфікаційній роботі була розроблена веб-система цифрового супроводу реабілітації та медичного контролю пацієнтів «MedAlert», призначена для організації взаємодії між лікарем та пацієнтом, контролю виконання медичних призначень, моніторингу стану здоров’я та супроводу реабілітаційного процесу. Реалізовані механізми формування індивідуальних планів реабілітації, збору та аналізу медичних показників, оцінювання рівня комплаєнсу та автоматичного клінічного тріажу дозволяють підвищити ефективність спостереження за пацієнтами, своєчасно виявляти потенційні ризики та підтримувати процес прийняття рішень медичним персоналом.

Для досягнення поставленої мети був проведений аналіз сучасних інформаційних систем у сфері охорони здоров’я показав, що більшість існуючих рішень орієнтовані або на ведення медичної документації, або на дистанційні консультації, тоді як питання контролю виконання індивідуальних реабілітаційних програм, моніторингу стану пацієнта та своєчасного інформування лікаря потребують подальшого розвитку.

У результаті дослідження було сформульовано функціональні та нефункціональні вимоги до веб-системи підтримки процесу реабілітації пацієнтів. На основі проведеного аналізу спроєктовано структуру бази даних, визначено взаємозв’язки між основними сутностями системи та розроблено архітектуру програмного рішення з використанням сучасних вебтехнологій.

У практичній частині роботи реалізовано веб-систему «MedAlert», побудовану за клієнт-серверною архітектурою з використанням React, TypeScript, Supabase та PostgreSQL. Реалізовано механізми автентифікації та розмежування доступу між ролями лікаря і пацієнта, що забезпечує безпечну роботу користувачів у межах власних функціональних можливостей.

У системі реалізовано модулі керування профілями користувачів, створення та супроводу індивідуальних планів реабілітації, ведення історії активності пацієнта, зберігання медичних документів, моніторингу виконання призначень та

перегляду результатів спостереження. Лікар отримує можливість формувати персоналізовані програми реабілітації, контролювати їх виконання та аналізувати динаміку стану пацієнтів.

Важливою складовою розробленої системи є реалізація механізму автоматичного розрахунку комплаєнсу, який дозволяє оцінювати рівень прихильності пацієнта до виконання призначень. Отримані показники використовуються для формування аналітичних звітів і виявлення пацієнтів із підвищеним ризиком порушення режиму лікування.

Крім того, у системі реалізовано модуль клінічного тріажу, який виконує автоматичний аналіз внесених пацієнтом фізіологічних показників та класифікує їх за рівнями ризику. У разі виявлення відхилень від встановлених норм система формує відповідні сповіщення для лікаря, що дозволяє оперативно реагувати на потенційно небезпечні зміни стану здоров’я пацієнта.

Проведене тестування підтвердило працездатність основних функціональних модулів системи та коректність взаємодії між лікарем і пацієнтом у межах основних сценаріїв використання. Результати апробації показали можливість застосування розробленого програмного продукту для організації цифрового супроводу реабілітаційного процесу, моніторингу виконання призначень та підтримки прийняття рішень лікарем.

Практична цінність розробленої системи полягає у підвищенні ефективності взаємодії між лікарем і пацієнтом, автоматизації контролю виконання рекомендацій, централізованому зберіганні медичної інформації та своєчасному виявленні ризикових ситуацій під час проходження реабілітації.

Подальший розвиток системи може бути пов’язаний із впровадженням мобільного застосунку, інтеграцією з медичними інформаційними системами та електронними медичними картками, використанням алгоритмів машинного навчання для прогнозування ризиків, а також розширенням можливостей автоматизованого клінічного моніторингу пацієнтів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Міністерство охорони здоров'я України. Офіційний вебсайт. URL : https://moz.gov.ua (дата звернення: 17.05.2026).

2. World Health Organization. Ukraine emergency situation. URL : https://www.who.int/europe/emergencies/situations/ukraine-emergency (accessed: 17.05.2026).

3. National Health Service. Digital transformation in health care URL : https://www.england.nhs.uk/digitaltechnology/ (accessed: 17.05.2026).

4. European Commission. Digital health and care URL : https://health.ec.europa.eu/ehealth-digital-health-and-care_en (accessed: 17.05.2026).

5. World Health Organization. Adherence to long-term therapies: evidence for action. URL : https://www.who.int/chp/knowledge/publications/adherence_full_report.pdf (accessed: 17.05.2026).

6. World Economic Forum. Digital health platforms and future healthcare. URL : https://www.weforum.org/agenda/archive/digital-health/ (accessed: 17.05.2026).

7. Офіційний веб-портал електронної медичної системи Helsi для пацієнтів та лікарів. URL :https://helsi.me/about (дата звернення: 17.05.2026).

8. About Health Portal – Terviseportaal // Estonian Health Insurance Fund / Health and Welfare Information Systems Centre (TEHIK). 2024. URL : https://www.terviseportaal.ee/en/terviseportaalist (date of access: 17.05.2026).

9. Germany changes rules for Digital Health Applications (DiGA) – More Reporting, More Transparency, More Work // Inside EU Life Sciences. 2026. URL : https://www.insideeulifesciences.com/2026/02/03/germany-changes-rules-for-digital-health-applications/ (date of access: 17.05.2026).

10. DiGA Directory | All Prescription Apps in Germany // BfArM DiGA-Verzeichnis. 2026. URL : https://www.diga-verzeichnis.de/en (date of access: 17.05.2026).

11. DyCare. ReHub Telerehabilitation. URL : https://www.dycare.com/en/rehub-telerehabilitation (accessed: 17.05.2026).

12. Weldring T. Patient-Reported Outcomes in Clinical Practice / T. Weldring, S. M. Smith // Federal Practitioner. 2013. Vol. 30, № 12. P. 8–12.

13. National Early Warning Score (NEWS) 2: Standardising the assessment of acute-illness severity in the NHS // Royal College of Physicians. London : RCP, 2017. 32 p.

14. Saripalle R. HL7 FHIR: An introduction to the Companion Health Cloud / R. Saripalle, Runyan C. // *Journal of Biomedical Informatics*. 2019. Vol. 94. P. 103-114.

15. European Commission. Data protection rules URL : https://commission.europa.eu/law/law-topic/data-protection_en (accessed: 17.05.2026).

16. Регламент Європейського Парламенту і Ради (ЄС) 2016/679 від 27 квітня 2016 року про захист фізичних осіб у зв'язку з опрацюванням персональних даних і про вільний рух таких даних, та про скасування Директиви 95/46/ЄС (Загальний регламент про захист даних) URL : https://zakon.rada.gov.ua/laws/show/984_008-16 (дата звернення: 18.05.2026).

17. Про захист персональних даних : Закон України від 01.06.2010 № 2297-VI URL : https://zakon.rada.gov.ua/laws/show/2297-17 (дата звернення: 18.05.2026).

18. Чорноус В. О., КузнєцоваО. В. Сучасні тенденції використання Front-end технологій при розробці веб-орієнтованих систем. *Комп'ютерні системи та мережі*. 2023. № 12. С. 45–52.

19. Гайнага В. А. Дослідження та розробка кросплатформенних веб-застосунків за технологією Progressive Web Apps (PWA). *Вісник інженерної академії*. 2024. № 2. С. 112–118.

20. Bradly J. Building Scalable Applications with Supabase and PostgreSQL / J. Bradly. San Francisco : TechPress, 2025. 264 p.