МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет

імені Петра Могили

Факультет комп'ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри,
д-р техн. наук, проф.

_____ І. М. Журавська

« __ » _____ 2024 р.

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

Інтелектуальна система моніторингу та діагностики раку молочної залози

Спеціальність 123 Комп'ютерна інженерія

123 – КМР.ПЗ.00 – 605.21810518

Студент _____С. В. Овчар
підпис
«__» _____ 202__ р.

Керівник
доц. каф. КІ, канд. техн. наук, доцент _____Я. М. Крайник
підпис
«__» _____ 202__ р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Зав. кафедри _____ І. М. Журавська

«_____» _____ 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної магістерської роботи

Видано студенту групи 605м факультету комп'ютерних наук

Овчару Сергію Віталійовичу
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи
Інтелектуальна система моніторингу та діагностики раку молочної залози

Затверджена наказом по ЧНУ ім. Петра Могили від 08.11.2023 № 226.

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Очікуваним результатом роботи є комплекс програмного забезпечення, що здійснює часткову автоматизацію збору даних про стан пацієнтів для діагностики та моніторингу раку молочної залози. Вхідними даними є перелік вимог до функціоналу програмного забезпечення.

4. Перелік питань, що підлягають розробці
1) огляд сучасних підходів до діагностики раку молочної залози;
2) аналіз переваг та недоліків існуючих систем в сфері надання медичних послуг та контролю за здоров'ям;
3) розробка програмного забезпечення для автоматизації збору даних про стан пацієнта;
4) забезпечення комунікації з датчиками для автоматизованого збору медичних даних пацієнта;

5. Перелік графічних матеріалів

слайди презентації

6. Завдання до спеціальної частини _
Проведення оцінки умов праці фахівців підприємства, а саме аналіз виробничого приміщення, оцінка природного освітлення. Визначення рівня електричної та пожежної безпеки підприємства

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Д-р біол. наук, професорка Григор'єва Л. І.	Кафедра екології Навчально-наукового медичного інституту ЧНУ ім. Петра Могили	Спеціальна частина з охорони праці та безпеки у надзвичайних ситуаціях

Керівник роботи
доцент кафедри. КІ, канд. техн. наук, доцент Крайник Ярослав Михайлович
(посада, прізвище, ім'я, по батькові)

_____
(підпис)

Завдання прийнято до виконання

Овчар Сергій Віталійович
(прізвище, ім'я, по батькові студента)

_____
(підпис)

Дата видачі завдання «____» _____ 20_____ р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної магістерської роботи

Тема:_ Інтелектуальна система моніторингу та діагностики раку молочної залози__

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КМР	01.09.2023	15.09.2023	*Виконано*
2.	Огляд літератури за темою роботи	15.09.2023	01.10.2023	*Виконано*
3.	Складання календарного плану КМР	01.10.2023	03.10.2023	*Виконано*
4.	Аналіз предметної області	05.10.2023	25.10.2023	*Виконано*
5.	Формування вимог до комплексу ПЗ	25.10.2023	10.11.2023	*Виконано*
6.	Розробка програмного комплексу	10.11.2023	10.12.2023	*Виконано*
7.	Розробка системи комунікації з датчиками	10.12.2023	20.12.2023	*Виконано*
8.	Тестування комплексу ПЗ	20.12.2023	05.01.2024	*Виконано*
9.	Аналіз результатів тестування	05.01.2024	10.01.2024	*Виконано*
10.	Відгук керівника КМР	10.01.2024	05.02.2024	*Виконано*
11.	Оформлення КМР та презентації	07.02.2024	08.02.2024	*Виконано*
12.	Попередній захист	31.01.2024	14.02.2024	*Виконано*
13.	Рецензування	15.02.2024	20.02.2024	*Виконано*
14.	Захист кваліфікаційної роботи	26.02.2024	27.02.2024	*Виконано*

Розробив здобувач ВО _____Овчар Сергій Віталійович_____ _____

(*прізвище, ім'я, по батькові)* *(підпис)*

«__» _____ 20__ р.

Керівник роботи доц. каф. КІ _Крайник Ярослав Михайлович_ _____

(підпис)

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної магістерської роботи

«Інтелектуальна система моніторингу та діагностики раку молочної залози»

студент групи 605м Овчар Сергій Віталійович

Керівник: канд. техн. наук, доцент Крайник Ярослав Михайлович

Рак молочної залози становить значну загрозу здоров'ю жінок, і його рання діагностика є ключовим фактором успішного лікування. За даними Всесвітньої організації здоров'я, раннє виявлення раку молочної залози може значно знизити смертність та покращити прогнози лікування.

Актуальність теми кваліфікаційної роботи полягає у потребі покращення методів діагностики одного з найпоширеніших онкологічних захворювань у світі. Використання інтелектуальних систем діагностики обіцяє значно підвищити швидкість та зручність діагностики, забезпечуючи більш раннє виявлення захворювань. Інновації у цій області мають потенціал революціонізувати медичну практику, забезпечуючи високу чутливість та специфічність в діагностиці.

Об’єкт дослідження (розробки): Використання інтелектуальних систем для автоматизації медичних обстежень.

Предмет дослідження (розробки): Мобільний застосунок з можливістю комунікації з медичними датчиками та автоматичного оновлення даних про стан пацієнтів.

Мета роботи: розробити інтелектуальну систему для ранньої діагностики раку молочної залози з використанням мобільного застосунку для пацієнтів з можливістю підключення датчиків для оновлення даних про стан пацієнтів в режимі реального часу.

Для досягнення мети було поставлено наступні завдання:

– провести огляд сучасних методів діагностики раку молочної залози;

– оглянути сучасні інтелектуальні системи в сфері надання медичних послуг;

– проаналізувати переваги та недоліки існуючих систем;

– розробити проєкт мобільного застосунку для пацієнтів, що надає можливість частково автоматизувати процес діагностики;

– продемонструвати практичну здійсненність та перспективність використання мобільних застосунків в сфері надання медичних послуг.

Робота складається з переліку скорочень, вступу, чотирьох розділів, висновку, переліку джерел посилання та трьох додатків.

У вступі наводяться ключові аргументи, що підкреслюють важливість обраної теми, визначаються об'єкт та предмет дослідження, ціль дослідження та завдання, які необхідно вирішити для досягнення мети роботи.

В першому розділі роботи здійснено огляд сучасних методів ранньої діагностики раку молочної залози та проведено аналіз систем, що використовуються для автоматизації в сфері надання медичних послуг.

Другий розділ присвячений детальному опису програмного та апаратного забезпечення використаного для розробки.

У третьому розділі описано технології для налаштування комунікації програмного комплексу з датчиками, що автоматично передають медичні дані пацієнтів для подальшого аналізу.

У четвертому розділі описано архітектуру та процес розробки системи.

У висновку подано огляд результатів виконання кваліфікаційної роботи.

Додатки містять код програмного забезпечення та інформацію про апробацію роботи.

Спеціальна частина з охорони праці розглядає дотримання норм охорони праці на робочому місці.

Кваліфікаційна робота містить 66 сторінок (без додатків), 30 рисунків, 48 джерел посилання та 3 додатки.

ABSTRACT

of the master's thesis

«Smart System for Breast Cancer Monitoring and Diagnosis»

Student: Serhii Ovchar

Supervisor: Cand. Sc. (Tech.), Assoc. Prof. Yaroslav Krainyk

Breast cancer poses a significant health threat to women, and its early diagnosis is a crucial factor for successful treatment. According to the World Health Organization, early detection of breast cancer can substantially reduce mortality and improve treatment outcomes.

The relevance of this thesis topic lies in the critical need to improve diagnostic methods for one of the most common oncological diseases globally. The use of intelligent diagnostic systems promises to significantly enhance the speed and convenience of diagnostics, enabling earlier disease detection. Innovations in this field have the potential to revolutionize medical practice, providing high sensitivity and specificity in diagnostics.

The object of the research is the use of intelligent systems for automation of various medical examinations.

The subject of the research is a mobile application capable of communicating with medical sensors and automatically updating patient health data.

Objective of the research is to develop an intelligent system for the early diagnosis of breast cancer using a mobile application with the capability to connect sensors for real-time health data updates.

To accomplish the objective of the study, the following tasks were established:

1) Conduct a review of modern methods for early diagnosis of breast cancer;
2) Examine current intelligent systems in the provision of medical services;
3) Analyze the advantages and disadvantages of existing systems;

4) Develop a project for a patient mobile application that allows partial automation of the diagnostic process;
5) Demonstrate the practical feasibility and prospects of using mobile applications in medical services.

The work consists of a list of abbreviations, an introduction, four chapters, a conclusion, a list of reference sources and three appendices.

The introduction presents key arguments that underscore the importance of the chosen topic, defines the research object and subject, the aim of the study, and the tasks required to achieve the thesis's goal.

First chapter of the study reviews modern methods for the early diagnosis of breast cancer and analyzes systems currently used for automation of various medical services.

Second chapter is dedicated to a detailed description of the software development technologies used in this study.

Third chapter describes the technologies used for communicating with sensors that automatically transmit patient medical data for further analysis.

Fourth chapter describes the process of software complex development and the architecture of the system.

The conclusion provides an overview of the results of the qualification work.

Appendices include software code and information on the approbation of the study.

Special part on labor protection provides an overview of the occupational safety standards at the workplace.

This qualification work contains 66 pages (excluding appendices), 30 figures, 48 reference sources, and 3 appendices.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

IMT	–	Індекс маси тіла
MPT	–	Магнітно-резонанса томографія
AOT	–	(англ. ahead-of-time) - компіляція перед виконанням
API	–	(англ. Application Programming Interface) інтерфейс програмування застосунків
ARM	–	(англ. Advanced RISC Machine) - 32-бітна RISC архітектура процесорів
BaaS	–	(англ. Backend as a Service) - модель надання послуг, що передбачає можливість легкої інтеграції хамрних сховищ та API
DCE-MRI	–	(англ. Dynamic Contrast Enhanced Magnetic Resonance Imaging) — контрастно- підсилена магнітно-резонанса томографія
ISO	–	(англ. International Organization for Standardization) - Міжнародна організація зі стандартизації
LLVM	–	(англ. Low Level Virtual Machine) - універсальна система аналізу, трансформації і оптимізації програм
mHealth	–	(англ. Mobile Health) - практикування медицини з використанням мобільних пристроїв
MRI	–	(англ. Magnetic Resonance Imaging) — Магнітно-резонанса томографія
NDK	–	(англ. Native Development Kit) — набір інструментів для розробки компонентів програмного забезпечення для платформи Android
SDK	–	(англ. Software Development Kit) — набір із засобів розробки
UI	–	(англ. User Interface) - інтерфейс користувача

ВСТУП

Рак молочної залози становить значну загрозу здоров'ю жінок, і його рання діагностика є ключовим фактором успішного лікування. За даними Всесвітньої організації здоров'я, раннє виявлення раку молочної залози може значно знизити смертність та покращити прогнози лікування [1].

Актуальність теми кваліфікаційної роботи полягає у критичній потребі покращення методів діагностики одного з найпоширеніших онкологічних захворювань у світі. Використання інтелектуальних систем діагностики обіцяє значно підвищити швидкість та зручність діагностики, забезпечуючи більш раннє виявлення захворювань. Інновації у цій області мають потенціал революціонізувати медичну практику, забезпечуючи високу чутливість та специфічність в діагностиці. Автоматизація процесу діагностики може зменшити навантаження на медичний персонал та скоротити час очікування на результати досліджень [2].

Об’єкт дослідження (розробки): Використання інтелектуальних систем для автоматизації медичних обстежень.

Предмет дослідження (розробки): Мобільний застосунок з можливістю комунікації з медичними датчиками та автоматичного оновлення даних про стан пацієнтів.

Мета роботи: розробити інтелектуальну систему для ранньої діагностики раку молочної залози з використанням мобільного застосунку для пацієнтів з можливістю підключення датчиків для оновлення даних про стан пацієнтів в режимі реального часу.

Для досягнення мети було поставлено наступні завдання:

– провести огляд сучасних методів діагностики раку молочної залози;

– оглянути сучасні інтелектуальні системи в сфері надання медичних послуг;

– проаналізувати переваги та недоліки існуючих систем;

– розробити проєкт мобільного застосунку для пацієнтів, що надає можливість частково автоматизувати процес діагностики;

– продемонструвати практичну здійсненність та перспективність використання мобільних застосунків в сфері надання медичних послуг.

Методи дослідження: аналіз і синтез, порівняння, комп’ютерне моделювання.

Практичне значення результатів роботи полягає в зменшенні адміністративного тиску на медичний персонал за допомогою автоматизації певних діагностичних досліджень.

Робота пройшла **апробацію** на Міжнародній науково-практичній конференції «Інформаційні технології і автоматизація – 2023» (м. Одеса, 19-20 жовтня 2023 р.).

Публікації: Основні положення та результати роботи було опубліковано у збірнику тез конференції «Інформаційні технології і автоматизація – 2023» [3].

1 ОГЛЯД СУЧАСНИХ МЕТОДІВ ДІАГНОСТИКИ ТА АВТОМАТИЗАЦІЇ НАДАННЯ МЕДИЧНИХ ПОСЛУГ. ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Огляд сучасних методів ранньої діагностики раку молочної залози

Найбільш розповсюджені в сучасній практиці методи діагностики раку молочної залози включають кілька ключових технік [4]:

- мамографія;
- магнітно-резонансна томографія (МРТ);
- контрастно-підсилена МРТ (DCE-MRI)
- біопсія.

Мамографія - це рентгенівський знімок молочної залози, який може виявити доброякісні або злоякісні зміни. Мамографію можна використовувати як для скринінгу, так і для діагностики [5].

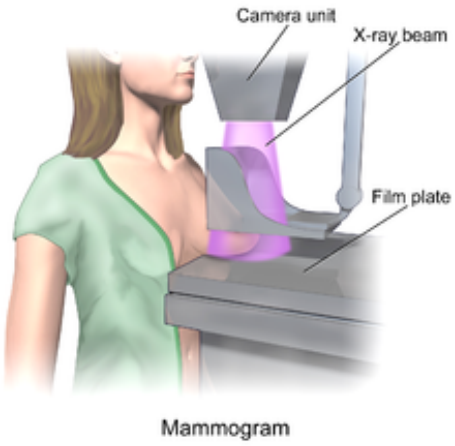


Рисунок 1.1 – Мамографія

Мамографія для скринінгу проводиться в спробі виявити будь-які ранні ознаки раку молочної залози, навіть до появи симптомів, для зниження смертності за рахунок раннього діагностування.

МРТ молочної залози - це ненавантажувальний і неіонізуючий діагностичний засіб зображення, який використовує радіочастотні хвилі

низької енергії та магнітне поле для отримання детальних зображень структури молочної залози. МРТ має здатність виявляти підозрілі злоякісні новоутворення молочної залози, які часто уникають клінічного та мамографічного виявлення [6].

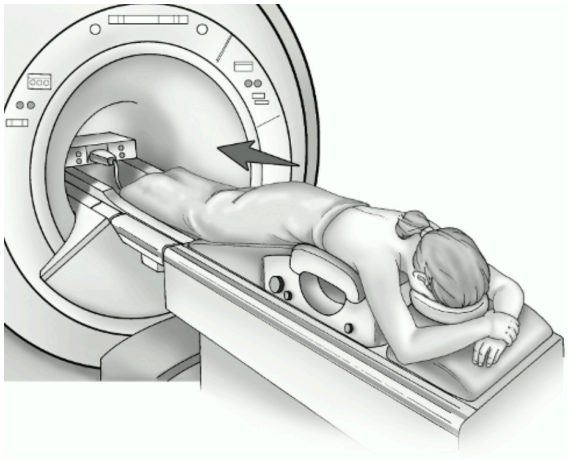


Рисунок 1.2 – Проведення МРТ молочних залоз

Через високу чутливість і нижчу специфічність МРТ молочної залози вона широко використовується в діагностиці раку молочної залози, що призводить до знаходження мас, що не були виявлені в результаті інших видів досліджень [7].

Динамічна контрастно-підсилена МРТ молочної залози працює за допомогою внутрішньовенного введення парамагнітного контрастного агента. Ця методика кількісно визначає ступінь васкуляризації тканини, склад міжклітинного простору та диференціацію змін тканини. Методика динамічної контрастно-підсиленої МРТ є ненавантажувальною і тривимірною, що дозволяє візуалізувати ступінь захворювання до морфологічних змін та допомагає прогнозувати загальну відповідь або до початку терапії, або на ранньому етапі лікування [8,9].

Єдиним визначним методом діагностики раку молочної залози є біопсія молочної залози. Для підвищення діагностичної точності та усунення якомога більшої кількості помилково-негативних результатів одночасно виконуються декілька видів обстежень, включно з біопсією [10].

Вищенаведені методи досліджень зазвичай базуються на фізичному огляді та діагностичних зображеннях, що потребує виділення значної кількості часу від пацієнтів та зусиль з боку медичного персоналу для проведення трудомістких діагностичних досліджень.

1.2 Сучасні алгоритми автоматизованого аналізу наявності раку молочної залози

Сучасні дослідження показують, що існують нові способи ранньої діагностики раку молочної залози, які можуть бути частково автоматизовані при збереженні високої точності діагностики [11].

В результаті досліджень було виявлено, що автоматизовані дослідження антропометричних вимірів та даних крові пацієнтів можуть з достатньо високою точністю прогнозувати наявність раку молочної залози без використання традиційних методів діагностики [12]. Найбільш важливими показниками для аналізу є вік, індекс маси тіла, рівень глюкози та резистину в крові.

Резистин – це нещодавно відкритий білок зі специфічними для тканин моделями вираження. Хоча два інших білки, відомі як молекули, подібні до резистину (RELM), та білки, знайдені в запальній зоні (FIZZ), були описані, резистин є єдиним з вищенаведеного переліку білком, який секретує адипоцити – жирові клітини, здатні накопичувати та перетворювати пухку сполучну тканину на жир [13].

На основі дослідження [12], резистин, вік, індекс маси тіла (ІМТ) та рівень глюкози використовуються як показники у моделі для прогнозування раку молочної залози. Кожен з цих факторів може бути пов'язаний з ризиком раку: резистин асоціюється з запаленням, ІМТ з гормональними порушеннями, вік є відомим фактором ризику, а рівень глюкози може вказувати на метаболічні зміни, пов'язані з наявністю раку молочної залози.

Використання цього методу дослідження не потребує проведення складних діагностичних процедур. Збір деяких даних – індексу маси тіла,

рівню глюкози в крові – може бути автоматизоване, що зменшує навантаження як на пацієнта, так і на медичний персонал.

1.3 Огляд мобільних застосунків в сфері надання медичних послуг

Проведення попереднього аналізу існуючих рішень та технологій є критично важливим етапом в процесі розробки. Цей аналіз дозволяє визначити потенціал нового продукту, уникнути повторення існуючих помилок та сприяє створенню інноваційного та конкурентоспроможного продукту.

Аналіз існуючих технологій допомагає виявити найсучасніші тенденції та напрямки розвитку. Це також включає оцінку можливостей та обмежень різних технологій, що дозволяє вибрати найбільш оптимальні рішення для реалізації конкретного проекту.

Після короткочасного використання технологій, характерним для багатьох користувачів мобільних застосунків та нових технологій є припинення користування ними [14]. Це підкреслює важливість створення інтуїтивно зрозумілих, корисних та зручних у використанні продуктів, які відповідають очікуванням користувачів, вказує на необхідність глибшого аналізу потреб та вимог кінцевих користувачів. Успішні технологічні рішення повинні фокусуватися не тільки на інноваційній складовій, але й на створенні позитивного користувацького досвіду, що спонукає до тривалого використання будь-якої технології.

Згідно з останніми дослідженнями в сфері mHealth [15], більше 80% mHealth- проектів розроблюються для використання з смартфонами, а близько 75 відсотків з усіх проектів мають власні мобільні застосунки. Більшість проектів також покладаються на використання сенсорів, іноді – використання wearable пристроїв.

Помітна увага в сучасних mHealth-проектах приділяється захворюванням, що впливають на якість життя – діабету, депресії, біполярному розладі, хронічним захворюванням [15]. Захворювання та стани відображають широкий спектр медичних проблем, з якими сучасні

технологічні розробки у сфері мобільного здоров'я намагаються впоратися, пропонуючи новітні рішення та підходи. Велика увага також приділяється допомозі користувачам у веденні здорового способу життя.

Цифрові платформи все частіше використовуються для надання лікарям швидкого доступу до величезної кількості медичної та лікарської інформації. Фахівці часто є учасниками онлайн-спільнот, і таким чином передають цінні знання та досвід, що може допомогти покращити якість інформації про здоров'я, доступної онлайн [16].

MyChart - це комплексний застосунок для охорони здоров'я, який призначений для спрощення взаємодії пацієнтів з їхніми медичними записами та лікарями. Розроблений компанією Epic Systems, лідером у сфері медичного програмного забезпечення, MyChart вирізняється як один з найбільш широко використовуваних порталів у Північній Америці [17].

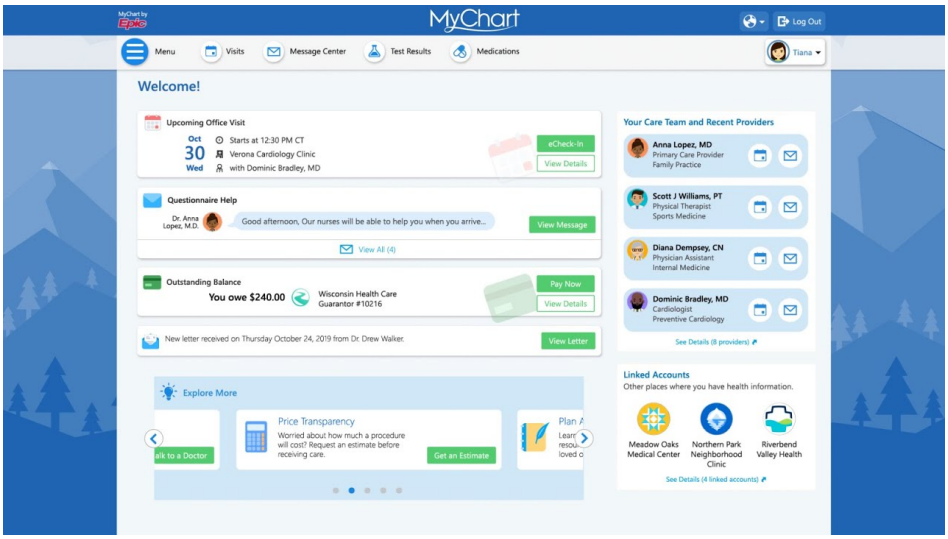


Рисунок 1.3 – Інтерфейс MyChart

Основною функцією MyChart є надання пацієнтам безпосереднього доступу до їхніх електронних медичних записів. Користувачі можуть переглядати свою медичну історію, записи про вакцинацію, результати лабораторних аналізів та радіологічні зображення. Цей доступ дає пацієнтам можливість бути більш обізнаними щодо свого здоров'я та лікування.

Застосунок також дозволяє користувачам записуватися на прийоми до медичних працівників. Крім того, він пропонує функцію перепланування та скасування, що робить управління прийомами високоефективним.

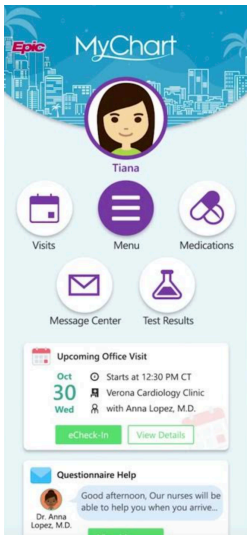


Рисунок 1.4 – Інтерфейс мобільного застосунку MyChart

MyChart надає безпечну платформу для обміну повідомленнями, де пацієнти можуть безпосередньо спілкуватися зі своїми лікарями або медичною командою. Ця функція є важливою для швидких консультацій, запитань щодо подальших дій та отримання медичних порад без необхідності особистого візиту.

Пацієнти можуть переглядати свої поточні рецепти, запитувати повторні рецепти та отримувати інструкції щодо прийому ліків.

MyChart може інтегруватися з деякими медичними пристроями, дозволяючи автоматично оновлювати дані про здоров'я, такі як кількість кроків і частота серцевих скорочень [18].

Загалом, MyChart – це потужний інструмент, який значно покращив спосіб взаємодії пацієнтів з їхніми лікарями та контроль за станом здоров’я Оскільки медична індустрія продовжує еволюціонувати, ймовірно, що MyChart також буде адаптуватися та удосконалюватися, далі закріплюючи свою роль як важливого компоненту сучасної медицини.

Застосунок SkinVision – це цифровий медичний інструмент, розроблений для допомоги у ранньому виявленні раку шкіри. Він дозволяє користувачам фотографувати плями на шкірі та отримувати оцінку ризику за допомогою алгоритмів штучного інтелекту.

Рисунок 1.5 – Інтерфейс мобільного застосунку SkinVision

SkinVision стверджує точність у виявленні ознак раку шкіри в 95% [19]. Проведені дослідження точності виявлення вказують на загальну чутливість у 73%, специфічність 83%, і точність 81% [20].

Застосунок зручний у використанні, дозволяє користувачам фотографувати свої шкірні плями та отримувати швидкі оцінки ризику, часто протягом 30 секунд. SkinVision служить попередньою перевіркою, потенційно зменшуючи непотрібні візити до медичних фахівців.

У доповнення до обов’язкових сертифікацій для медичних пристроїв, SkinVision отримав добровільні сертифікати ISO 13485 та ISO 27001, для забезпечення безпеки та захисту своїх медичних та інформаційних систем [21].

SkinVision є цінним інструментом у боротьбі з раком шкіри, пропонуючи зручний спосіб моніторингу стану шкіри. Однак важливо використовувати його як допоміжний засіб у поєднанні з консультаціями спеціалістів, особливо у випадках оцінки високого ризику.

Health — це програма для мобільних пристроїв, призначена для управління інформацією у сфері охорони здоров'я, яку представила компанія Apple Inc. 2 червня 2014 року під час Всесвітньої конференції розробників (WWDC). Цей застосунок встановлюється на iPhone та iPod Touch, що працюють на операційній системі iOS 8 або новіших версіях.

Програма забезпечує зберігання та відображення інформації про стан здоров'я користувача, включаючи показники артеріального тиску та рівень цукру в крові. Крім того, вона може відстежувати різні види даних, наприклад, кількість пройдених кроків. Застосунок здатен синхронізуватися з різними пристроями, такими як фітнес-браслети, розумні годинники та розумні ваги, для отримання необхідних даних [22].

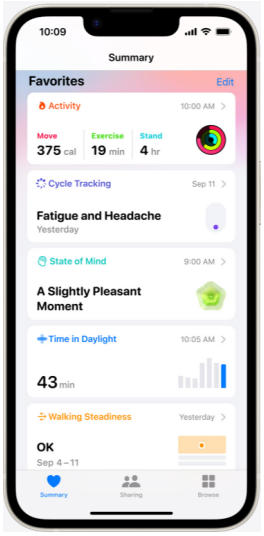


Рисунок 1.6 – Інтерфейс Apple Health

Одна з основних переваг Apple Health - це здатність консолідувати дані про здоров'я з різних джерел, включаючи iPhone, Apple Watch та сторонні застосунки. Такий централізований підхід забезпечує цілісний огляд метрик здоров'я користувача, таких як кроки, частота серцевих скорочень, шаблони сну та інше.

Apple Health інтегрується з Apple Watch, пропонуючи моніторинг різних метрик здоров'я в реальному часі, таких як частота серцевих скорочень,

кількість кроків і фізичні вправи. Ця інтеграція заохочує користувачів бути більш активними та уважними до свого здоров'я.

Apple Health підтримує введення даних з безлічі сторонніх застосунків. Це означає, що користувачі можуть відстежувати різні аспекти свого здоров'я, такі як дієта, сон і фізичні вправи, навіть якщо ці дані збираються іншими застосунками.

Користувачі можуть зберігати та переглядати свої медичні записи безпосередньо у застосунку. Ця функція неймовірно корисна для відстеження вакцинацій, результатів лабораторних аналізів, ліків та іншого.

Застосунок має інтуїтивно зрозумілий та чистий інтерфейс, що робить його легким у використанні для користувачів будь-якого рівня технічних навичок.

Apple приділяє велику увагу конфіденційності користувачів та безпеці даних. Дані про здоров'я шифруються та зберігаються в безпеці, що дає користувачам спокій стосовно їхньої особистої інформації про здоров'я [23].

В результаті проведення аналізу інтелектуальних систем в сфері надання медичних послуг, а також систем контролю за здоров’ям було виявлено певні недоліки та характерні особливості, що можуть бути покращені для надання приємного досвіду користування.

MyChart - це потужний інструмент, який значно покращив спосіб взаємодії пацієнтів з їхніми медичними працівниками та управління своїм здоров'ям. Хоча він перевершує у багатьох аспектах, завжди є простір для удосконалення, особливо в плані послідовності користувацького досвіду, безпеки даних та інклюзивності.

Досвід користування може відрізнятися в залежності від медичного закладу, оскільки різні установи можуть використовувати різні функції застосунку. Ця невідповідність може заплутати пацієнтів, які змінюють лікарів або використовують застосунок у різних медичних установах. Хоча MyChart і пропонує деяку інтеграцію з датчиками, він міг би далі розширити свою

сумісність з ширшим спектром пристроїв та застосунків для здоров'я і фітнесу, щоб надати користувачам всебічне управління та контроль над здоров'ям.

Як і будь-який застосунок, що обробляє чутливу інформацію про здоров'я, постійні зусилля щодо підвищення безпеки даних та конфіденційності є важливими. Користувачам потрібні постійні запевнення, що їхні дані про здоров'я захищені.

Розширення діапазону доступних мов та покращення функцій доступності для користувачів з інвалідністю може зробити MyChart більш інклюзивним.

SkinVision, хоча і надає оцінки ризику, не є заміною формального медичного діагнозу. Користувачам важливо розуміти, що застосунок надає лише настанови, а не остаточні діагнози. Точність застосунку, хоча і висока, не є непогрішною. Важливо пам'ятати, що жоден технологічний інструмент не може гарантувати 100% точності медичного діагнозу. Користувачам слід залишатися обережними та звертатися по професійну допомогу, особливо у випадках оцінки високого ризику. Мобільний застосунок SkinVision не надає користувачам консультації з фахівцями на основі автоматичного аналізу фото. Існує ризик, що користувачі можуть занадто покладатися на застосунок для виявлення раку шкіри. Це може призвести до затримки у наданні професійної консультації.

Найбільшим недоліком Apple Health є його обмеженість до iOS та пристроїв Apple. Ця ексклюзивність обмежує його використання лише користувачами Apple, залишаючи в стороні великий сегмент користувачів Android та інших мобільних операційних систем. Для оптимального використання Apple Health часто вимагається Apple Watch або інший пристрій, що може бути значним застосунковим витратами. Для людей з конкретними медичними станами, аналіз, який надає застосунок, може бути недостатньо всебічним, тому що основний фокус уваги в застосунку приділяється поверхневому огляду загального стану здоров’я. Для деяких користувачів

велика кількість доступних даних та метрик може бути складною для зрозумілого інтерпретування без медичної освіти. Безпосередній доступ медичних працівників до даних пацієнта, як і консультації фахівців, у Apple Health не надаються.

Висновки до розділу 1

Для інтелектуальної діагностики раку молочної залози було обрано алгоритм попередньої діагностики на основі показників віку, індексу маси тіла, рівня глюкози та концентрації резистину в крові, описаний в (1.2)

Враховуючи наведені у (1.3) недоліки та особливості використання проаналізованих систем, можна запропонувати декілька шляхів для їх вирішення:

– Для уникнення плутанини серед користувачів, які відвідують різні медичні установи, важливо стандартизувати основні функції застосунку. Це забезпечить послідовний досвід користування незалежно від медичного закладу.

– Збільшення сумісності з різними пристроями для моніторингу здоров'я та фітнес-застосунками може надати користувачам більш широкий спектр інструментів для контролю за своїм здоров'ям.

– Сертифікація системи захисту персональних та медичних даних, а також забезпечення неперервного оновлення заходів безпеки та прозорості у зберіганні та обробці даних про здоров'я є ключовим для збереження довіри користувачів.

– Розробка можливості онлайн-консультацій із фахівцями зможе допомогти користувачам уточнити ризики та отримати професійну пораду, зменшивши ризик використання застосунку як єдиного джерела діагностики.

– Створення версій застосунку для декількох операційних систем, підтримка різноманітних датчиків зробить застосунок доступним для більш широкого кола користувачів та допоможе уникнути надмірних витрат.

2 ПРОГРАМНО-АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ДІАГНОСТИКИ

Вибір правильних інструментів розробки програмного забезпечення є ключовим фактором успіху будь-якого проекту в області інформаційних технологій. Правильний набір інструментів не тільки спрощує процес розробки, але й підвищує продуктивність розробників, забезпечує вищу якість продукту та може значно скоротити час виведення продукту на ринок [24].

Інструменти розробки, які добре інтегруються з існуючими процесами та екосистемою, можуть значно підвищити продуктивність розробників. Автоматизація рутинних задач, таких як компіляція коду, тестування, відстеження помилок та розгортання, звільняє час розробників для зосередження на більш складних та творчих аспектах проекту. Крім того, інструменти, які пропонують чіткі та зручні інтерфейси для виконання цих задач, знижують поріг входження для новачків та сприяють швидшому впровадженню змін.

Якість програмного забезпечення безпосередньо залежить від здатності ідентифікувати та виправляти помилки на ранніх стадіях розробки. Використання інструментів статичного аналізу коду, автоматизованих систем тестування та систем управління версіями дозволяє систематично виявляти та усувати потенційні проблеми, перш ніж вони перетворяться на серйозні помилки. Це не тільки знижує витрати на підтримку та виправлення помилок після випуску продукту, але й покращує загальне враження користувачів від продукту.

Правильний вибір інструментів розробки може значно знизити загальні витрати на розробку продукту. Використання відкритого програмного забезпечення або інструментів з гнучкими ліцензійними умовами значно знижує витрати на придбання ліцензій [25]. Автоматизація процесів розробки знижує потребу в ручній праці та може скоротити час розробки, а отже, і витрати на зарплату команди. Крім того, вибір інструментів, які забезпечують

високу якість продукту та знижують ризик помилок, також знижує потенційні витрати на підтримку та виправлення помилок програмного забезпечення.

2.1 Flutter як інструмент для розробки крос-платформових мобільних застосунків

Крос-платформова розробка стає все більш популярною серед розробників та компаній, оскільки вона пропонує значні переваги порівняно з класичними інструментами розробки. Головною її перевагою є здатність використовувати єдину кодову базу для створення застосунків, що працюють на кількох платформах одночасно без необхідності одночасної підтримки декількох кодових баз [26]. Це не тільки зменшує час та витрати на розробку, але й спрощує процес підтримки та оновлення застосунків, оскільки зміни потрібно вносити лише в одному проекті. Велика кількість відкритих фреймворків розробки (див. рис. 2.1) дозволяє обрати інструмент, що найбільше задовольняє потреби розробників.

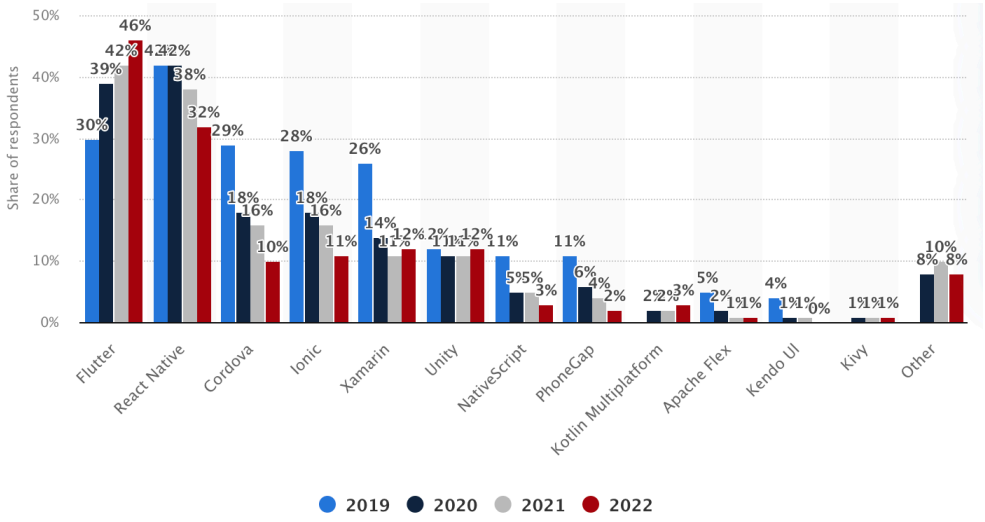


Рисунок 2.1 – Популярність фреймворків крос-платформової мобільної розробки [27]

Використання фреймворків, таких як Flutter або React Native, дозволяє розробникам створювати застосунки, що забезпечують багате та інтуїтивно зрозуміле користувацьке середовище, що не втрачає якості та не погіршує

досвід використання у порівнянні з застосунками, створеними класичним методом [26].

Серед доступних інструментів Flutter заслуговує особливої уваги через свої характеристики. Flutter є відкритим UI-фреймворком, створеним Google, який використовує Dart як свою основну мову програмування [28]. Вперше представлений у 2017 році, Flutter дозволяє розробникам створювати візуально привабливі, продуктивні застосунки для мобільних, веб- та десктопних платформ з єдиної бази коду.

Рисунок 2.2 – Логотип Flutter

Dart є мовою програмування з відкритим кодом, розробленою Google, яка вперше була представлена у 2011 році. Ця мова була створена з метою забезпечення більш продуктивного та гнучкого способу розробки сучасних веб- та мобільних застосунків [29]. Dart є об'єктно-орієнтованою, класовою мовою з C-подібним синтаксисом, що робить її знайомою та доступною для багатьох розробників. Однією з ключових особливостей Dart є її підтримка як для компіляції у JavaScript, так і для компіляції у нативний машинний код, що дозволяє використовувати її як для веб-розробки, так і для створення мобільних застосунків.

Рисунок 2.3 – Логотип Dart

Dart має високі оцінки за такими основними критеріями:

Одна з головних цінних пропозицій Flutter полягає в тому, що він економить інженерні ресурси, дозволяючи розробникам створювати програми

для iOS і Android з однаковою кодовою базою. Використання високопродуктивної мови ще більше прискорює розробників і робить Flutter більш привабливим. Більшість коду Flutter також побудовано на Dart.

Flutter дозволяє розробникам використовувати одну кодову базу для створення застосунків, які працюють як на iOS, так і на Android. Це не тільки спрощує процес розробки та тестування, але й значно знижує витрати на підтримку та оновлення застосунків. Flutter відрізняється від більшості інших варіантів створення мобільних застосунків, оскільки він не покладається ні на технологію веб-браузера, ні на набір віджетів, які постачаються з кожним пристроєм. Натомість Flutter використовує власний високопродуктивний механізм візуалізації для малювання віджетів [30].

Крім того, Flutter відрізняється тим, що він має лише тонкий шар коду C/C++. Flutter реалізує більшість своєї системи (компонування, жести, анімація, фреймворк, віджети тощо) у мові Dart, яку розробники можуть легко читати, змінювати, замінювати чи видаляти. Це дає розробникам контроль над системою, а також значно знижує планку доступності для більшості систем [30].

Flutter містить:

– Оптимізований механізм 2D-рендерінгу, перш за все для мобільних пристроїв;

– Набір віджетів, що реалізують дизайн і стиль Android та iOS;

– API для модульних та інтеграційних тестів;

– API взаємодії з нативними платформами та плагінів для підключення до системи та сторонніх SDK;

– Інструменти для тестування у Windows, Linux і Mac;

– Flutter DevTools (в поєднанні з Dart DevTools) для тестування, відлагодження та профілювання програм;

– Інструменти командного рядка для створення, побудови, тестування та компіляції програм.

Для запуску застосунків Flutter на Android код двигуна C і C++ компілюється за допомогою NDK Android. Код Dart (як SDK, так і власне код застосунку) компілюється перед виконанням (AOT) у власні бібліотеки, бібліотеки ARM і x86. Ці бібліотеки включено в проект Android, який відповідає за запуск застосунку, і все це вбудовано в .apk файл для встановлення на мобільному пристрої. Після запуску програма завантажує бібліотеку Flutter. Будь-яке відтворення, введення чи обробка подій делегується скомпільованому коду Flutter.

Для iOS, код двигуна C і C++ компілюється за допомогою LLVM. Код Dart (як SDK, так і власне код застосунку) компілюється перед виконанням (AOT) у бібліотеку ARM. Ця бібліотека включена в «Runner» проект iOS, і все це вбудовано в .ipa. Після запуску програма завантажує бібліотеку Flutter. Будь-яка візуалізація, введення чи обробка подій тощо делегуються скомпільованому коду Flutter.

Під час режиму відлагодження Flutter використовує віртуальну машину (Dart VM) для запуску свого коду, щоб увімкнути миттєве перезавантаження зі збереженням стану – функцію, яка дозволяє вносити зміни у запущений код без повторної компіляції. Під час роботи в цьому режимі в верхньому правому куті програми є видимий банер «Debug», який слугує нагадуванням про те, що продуктивність не є характерною для готової версії програми.

У березні 2021 року командою Flutter було проведено виміри розміу завантаження мінімального застосунку Flutter [31], укомплектованого та стисненого як APK, що складав приблизно 4,3 МБ для ARM32 і 4,8 МБ для ARM64 [30].

На ARM32 базовий функціонал у стиснутому вигляді займав приблизно 3,4 МБ, структура та код програми – приблизно 765 КБ, файл LICENSE мав розмір в 58 КБ, а необхідний код Java (classes.dex) — 120 КБ.

В ARM64 базовий функціонал у стиснутому вигляді займав приблизно 4,0 МБ, структура та код програми – приблизно 659 КБ, файл LICENSE мав розмір в 58 КБ, а необхідний код Java (classes.dex) – 120 КБ.

Ці цифри були виміряні за допомогою apkanalyzer, який також вбудований в Android Studio [32].

На iOS випуск IPA тієї ж програми має розмір завантаження 10,9 МБ. IPA є більшим, ніж APK, головним чином тому, що Apple шифрує двійкові файли в IPA, що робить стиснення менш ефективним [33].

Flutter використовує набір віджетів, які є основними будівельними блоками для створення інтерфейсів. Ці віджети можуть бути легко налаштовані та комбіновані для створення складних інтерфейсів. Ключовою особливістю Flutter є філософія "все є віджетом", де будь-які елементи інтерфейсу, такі як тексти та кнопки, а також більш складні структури, такі як анімації та жести, реалізовані як віджети.

Крім високої продуктивності та гнучкості у розробці інтерфейсів, Flutter також пропонує потужну систему анімацій, глибоку інтеграцію з існуючими кодовими базами, включаючи використання нативного коду, та підтримку спільноти та плагінів, що безперервно розширюється.

Завдяки підтримці механізму миттєвого перезавантаження «hot reload», Flutter дозволяє розробникам миттєво бачити результати змін у коді без необхідності повного перезапуску застосунку. Ця особливість значно прискорює процес розробки та спрощує експерименти з дизайном і функціоналом.

Як продукт Google, Flutter користується сильною підтримкою та великою спільнотою розробників. Це означає, що доступно багато ресурсів для навчання, велика кількість готових до використання плагінів та бібліотек, а також активний обмін знаннями та досвідом між розробниками.

Flutter підтримує широкий спектр платформ, запуск застосунків з яких можливий з одного проекту та одної кодової бази, використовуючи

розмежування коду Dart та нативного коду в різних директоріях (пакетах) проекту. Станом на 2024 рік, існує підтримка наступних платформ:

- Android версії 4.4 і вище (SDK 19+);
- iOS версії 12 і вище;
- Linux Debian версії 9 і вище;
- Linux Ubuntu версії 20.10 і вище (виключно 64bit);
- macOS версії 12 і вище;
- Windows версії 7 і вище;
- Web-браузери Chrome 96+, Firefox 99+, Safari 14+, Edge 96+.

Враховуючи вищенаведені характеристики фреймворку Flutter, доцільно обрати саме його для розробки клієнтської (мобільної) частити інтелектуальної системи для моніторингу та діагностики раку молочної залози.

2.2 Використання Backend-as-a-Service для розробки бізнес-логіки мобільних застосунків

Бекенд-частина системи, яку також називають серверною частиною, є критично важливим компонентом у структурі будь-якої інформаційної системи або веб-застосунку. Вона відповідає за обробку даних, зберігання інформації, аутентифікацію користувачів, авторизацію та забезпечення безпеки. Бекенд виконує бізнес-логіку і є мостом між користувацьким інтерфейсом та базою даних або іншими службами зберігання даних.

Бекенд є основою будь-якого застосунку, що підтримує його функціональність і забезпечує виконання складних операцій. Він обробляє запити від користувачів, виконує логіку застосунку, інтегрується з іншими службами та базами даних, та повертає результати на фронтенд для відображення користувачам. Без бекенду багато застосунків не могли б функціонувати, оскільки вони вимагають високої швидкості обробки даних, зберігання інформації на віддалених сховищах та інших функцій, що виконуються на сервері.

Бекенд складається з декількох ключових компонентів, включаючи сервери, бази даних, серверні застосунки та API [34].

Сервер – це фізичний або віртуальний комп'ютер, який обробляє запити від користувачів і повертає відповіді.

Бази даних зберігають великі обсяги інформації, які застосунок може запитувати, оновлювати, видаляти або додавати.

API дозволяє різним застосункам спілкуватися між собою, надаючи стандартизований метод для обміну даними.

Серверні застосунки виконують бізнес-логіку і оброблюють запити, використовуючи різноманітні програмні мови, такі як Java, Python, Ruby, Javascript та інші.

Розробка бекенду супроводжується певними викликами, включаючи забезпечення масштабованості, безпеки, і високої доступності системи. Масштабованість вимагає від системи здатності ефективно обробляти зростаючий обсяг запитів, тоді як безпека є ключовим аспектом для захисту даних від несанкціонованого доступу або втрати. Висока доступність гарантує, що застосунок залишається доступним для користувачів навіть у разі збоїв або підвищеного навантаження [34].

На сьогодні існує безліч інструментів розробки, які допомагають подолати початкову складність розробки, що є перевагою для стартапів, які прагнуть швидко вивести свій продукт на ринок. Послуга «Backend-as-a-Service» (BaaS) одним з найпопулярніших інструментів розробки програмного забезпечення для цього. Ця технологія спрощує налаштування застосунку і, зазвичай, працює шляхом передачі даних через власний API. Таким чином, розробники можуть зосередитись виключно на клієнтській частині застосунку та дизайні; неважливо, чи є фронтенд веб-базованим чи мобільним; єдина вимога для застосунку полягає в необхідності підключення до бекенд-системи через API для отримання та відправлення даних. Більшу частину повторюваної праці та шаблонного коду, пов'язаного зі створенням, читанням, оновленням

та видаленням даних, можна усунути, значно знижуючи витрати на розробку [35].

Backend as a Service (BaaS) є моделлю хмарного сервісу, яка дозволяє мобільним та онлайн-застосункам підключатися до хмарних бекенд-сервісів. До цієї категорії належать такі послуги, як зберігання даних, управління користувачами, зберігання файлів, геолокація, сповіщення, аналітика тощо [35]. Платформи BaaS, у своїй основі, дозволяють зберігати дані без проблем та витрат, пов'язаних зі створенням та підтримкою окремих сервісів для кожного застосунку. Швидка розробка є ключовим фактором, що сприяє зростанню ринку BaaS. Основні переваги BaaS включають прискорений вихід на ринок, зниження витрат на розробку та підвищену масштабованість.

Найбільш поширеними сервісами в сфері BaaS станом на сьогодні є Firebase та Supabase.

Firebase, розроблений і підтримуваний Google, є популярною платформою Backend as a Service (BaaS), яка надає широкий спектр інструментів і сервісів для розробки веб- та мобільних застосунків. Вона забезпечує розробників всім необхідним для швидкого створення, управління та масштабування застосунків, зосереджуючись на користувацькому досвіді без потреби займатися складними бекенд-процесами.

Рисунок 2.4 – Логотип Firebase

Основні характеристики та можливості Firebase [36]:

– Realtime Database & Firestore: Firebase пропонує дві бази даних: Realtime Database для синхронізації даних в реальному часі між користувацькими пристроями та Cloud Firestore, більш гнучку та

масштабовану версію бази даних, оптимізовану для більших застосунків.

- Authentication: Сервіс аутентифікації дозволяє легко інтегрувати різні методи входу в застосунок, включаючи електронну пошту, пароль, соціальні мережі та анонімну аутентифікацію.
- Cloud Functions: Ця можливість дозволяє виконувати серверний код на вимогу у відповідь на події, ініційовані Firebase аналітикою, аутентифікацією, входящими HTTPS-запитами та багатьма іншими джерелами.
- Hosting & Cloud Storage: Firebase надає послуги хостингу для статичного контенту (HTML, CSS, JavaScript, і т.д.) та Cloud Storage для зберігання файлів, таких як зображення, відео та інші великі об'єкти даних.
- Analytics: Інтегрована аналітика надає детальну інформацію про поведінку користувачів і дозволяє оптимізувати застосунок на основі зібраних даних.
- Push Notifications: Легко налаштовувані сповіщення дозволяють залучати користувачів, надсилаючи їм важливу інформацію або персоналізовані повідомлення.

Firebase став вибором для багатьох розробників завдяки своїй інтегрованості з іншими сервісами Google, включаючи Google Cloud Platform, та своїй здатності значно скоротити час та витрати на розробку.

Supabase, який часто називають відкритою альтернативою Firebase, є ще однією платформою Backend as a Service, яка пропонує набір інструментів для швидкої розробки веб- та мобільних застосунків.

Рисунок 2.5 – Логотип Supabase

Supabase зосереджений на наданні потужних SQL-засобів управління даними, забезпечуючи розробників потужним, але все ж таки простим у використанні інтерфейсом.

Основні характеристики та можливості Supabase [37]:

- PostgreSQL як база даних: Supabase використовує PostgreSQL, одну з найпопулярніших відкритих систем управління базами даних, що дозволяє розробникам використовувати потужні SQL-запити для роботи з даними.
- Realtime subscriptions: Аналогічно до Realtime Database у Firebase, Supabase дозволяє розробникам створювати застосунки, які можуть отримувати оновлення даних в реальному часі через WebSocket.
- Аутентифікація та авторизація: Supabase надає вбудовані рішення для аутентифікації користувачів, включаючи підтримку сторонніх провайдерів аутентифікації, таких як Google та GitHub.
- Storage: Для зберігання і обробки великих файлів, Supabase пропонує систему зберігання, яка дозволяє легко управляти файлами і забезпечує інтеграцію з базою даних.
- Повна інтеграція з PostgreSQL: Оскільки Supabase повністю сумісний з PostgreSQL, розробники можуть використовувати всі можливості цієї бази даних.

Supabase здобуває популярність серед розробників, які шукають гнучке, відкрите рішення для швидкої розробки застосунків з потужними можливостями управління даними. Його відкритий код та простота інтеграції з іншими інструментами роблять його привабливим вибором для проектів різного масштабу.

При оцінці бекенд-рішень вибір між Supabase та Firebase є поширеним дилемою, тому остаточний вибір правильного інструменту розробки потребує аналізу документації обох сервісів [38,39] та порівняння їх характеристик.

Supabase використовує PostgreSQL , дозволяючи повні можливості SQL та безпеку на рівні рядків. Firestore від Firebase є рішенням NoSQL з різними методами структурування та запитів даних. Підхід Supabase до управління даними є більш традиційним і орієнтованим на SQL, тоді як Firebase пропонує більш гнучку, документо-орієнтовану модель.

Обидві платформи пропонують надійні рішення для аутентифікації. Система аутентифікації на основі GoTrue від Supabase безпроблемно інтегрується з функціями безпеки PostgreSQL. Аутентифікація Firebase відома своєю простотою використання та інтеграцією з іншими сервісами Firebase.

Supabase розширює можливості PostgreSQL функціональністю оновлень в режимі реального часу, дозволяючи розробникам миттєво отримувати сповіщення про зміни в базі даних. Firebase також пропонує оновлення бази даних у режимі реального часу – функції, яка стала центральною для його популярності.

Відкритий код Supabase заохочує внески спільноти та прозорість. Firebase, який підтримується Google, має велику спільноту та обширну документацію, але не є відкритим джерелом.

Вибір між Supabase та Firebase часто зводиться до конкретних вимог проекту та переваг розробника. Фундамент Supabase на PostgreSQL може приваблювати тих, хто потребує потужності та гнучкості SQL, тоді як масштабована база даних NoSQL від Firebase та легкість інтеграції з сервісами Google можуть бути переважними для інших.

Одна з важливих переваг Supabase для сервісів в сфері контролю за здоров’ям пацієнтів є сумісність з вимогами Health Insurance Portability and Accountability Act (HIPAA) [40]. HIPAA встановлює критично важливі стандарти для захисту медичної інформації, вимагаючи від організацій впроваджувати комплексні заходи безпеки та процедури для забезпечення конфіденційності, цілісності та доступності медичних даних. Основні вимоги HIPAA можна розділити на кілька ключових областей [41]:

- Правило конфіденційності (Privacy Rule) встановлює стандарти для захисту медичної інформації пацієнтів. Воно вимагає від медичних установ та їхніх бізнес-партнерів використовувати розумні заходи безпеки для захисту конфіденційності і обмежує способи використання та розголошення такої інформації без згоди пацієнта.
- Правило безпеки (Security Rule) вимагає від організацій впроваджувати фізичні, адміністративні та технічні заходи безпеки для захисту електронної медичної інформації (ePHI). Це включає контроль доступу до інформації, аудитування дій з ePHI, управління цілісністю даних, а також використання захищених комунікаційних каналів.
- Правило передачі (Transactions and Code Sets Rule) стандартизує формати електронних трансакцій зі здоров'ям та медичними кодами, спрощуючи обмін даними між різними організаціями. Воно забезпечує, що всі медичні та страхові установи використовують єдині коди і стандарти для трансакцій, зменшуючи складність та витрати на обробку даних.
- Правило ідентифікаторів (Unique Identifiers Rule) встановлює вимоги до використання унікальних ідентифікаторів для медичних постачальників, планів медичного страхування та роботодавців, щоб забезпечити точне відстеження та ідентифікацію сторін у медичних трансакціях.
- Правило про повідомлення про порушення (Breach Notification Rule) зобов'язує організації повідомляти осіб, чия PHI була розкрита або втрачена внаслідок порушення безпеки, про таке порушення в певні терміни. Воно також вимагає повідомлення міністерства охорони здоров'я та соціальних служб та, у певних випадках, преси.

Серед сервісів, що пропонуються Supabase, для розробки проекту необхідно детально ознайомитися з функціями, що є ключовими для реалізації бекенд-частини системи – аутентифікації, зберігання даних в базі даних та створення API для комунікації з клієнтською частиною.

Supabase Auth є інтегрованим рішенням для аутентифікації та авторизації. Завдяки своїй тісній інтеграції з PostgreSQL, Supabase Auth дозволяє легко створювати контролб доступу до даних, забезпечуючи широкий спектр опцій аутентифікації включно з електронною поштою та паролем, а також підтримкою аутентифікації через зовнішні провайдери, таких як Google, GitHub та інші. Це дозволяє розробникам забезпечити користувачам зручний вхід в систему, використовуючи вже існуючі облікові записи, тим самим спрощуючи процес реєстрації та входу.

Окрім базової функціональності аутентифікації, Supabase Auth включає додаткові можливості для управління сесіями користувачів, відновлення забутих паролів, зміни паролів, а також детального налаштування правил доступу на рівні окремих користувачів або груп. Це забезпечує гнучкість та безпеку в управлінні доступом до ресурсів застосунку.

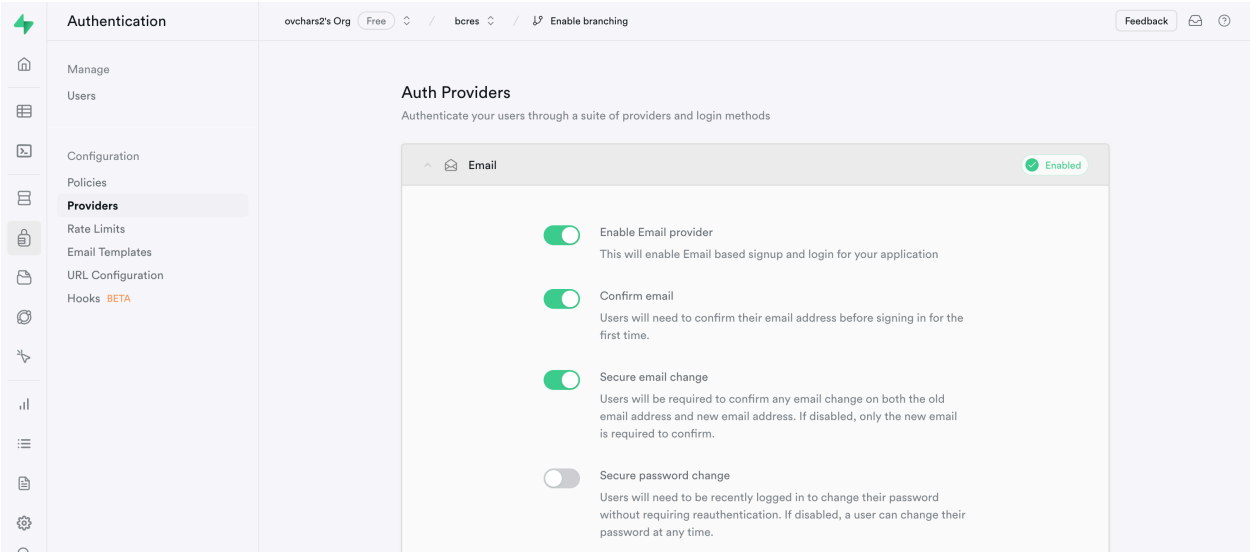


Рисунок 2.6 – Налаштування функції аутентифікації в Supabase

Supabase Database є потужним рішенням для управління даними, яке базується на PostgreSQL. Завдяки цьому, Supabase Database забезпечує

широкий спектр можливостей для роботи з даними, включаючи складні запити SQL, транзакції, підтримку індексів, зовнішніх ключів та т.ін.

Особливістю Supabase є його здатність надавати реальну синхронізацію даних в реальному часі, дозволяючи застосункам миттєво реагувати на зміни в базі даних без необхідності ручного оновлення або перезапитування даних. Це робить Supabase ідеальним вибором для застосунків, які вимагають високої інтерактивності та актуальності даних.

Крім того, Supabase надає розширені можливості для управління доступом та безпекою даних, дозволяючи розробникам детально налаштовувати права доступу на рівні рядків даних. Це забезпечує високий рівень контролю над тим, хто може переглядати, редагувати або видаляти інформацію в базі даних, що є критично важливим для застосунків, які обробляють конфіденційну або чутливу інформацію.

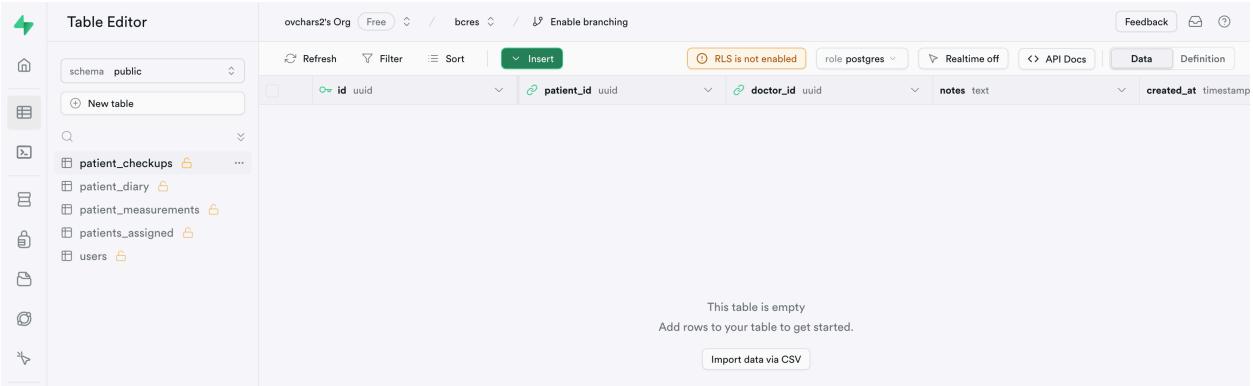


Рисунок 2.7 – Редактор таблиць Supabase

Supabase AI, що перебуває на стадії розробки та тестування, обіцяє трансформувати спосіб написання SQL запитів, дозволяючи користувачам формулювати запити природною мовою. Цей сервіс перетворює слова на складні SQL команди, значно спрощуючи доступ до даних та аналітику, навіть для неспеціалістів у галузі баз даних.

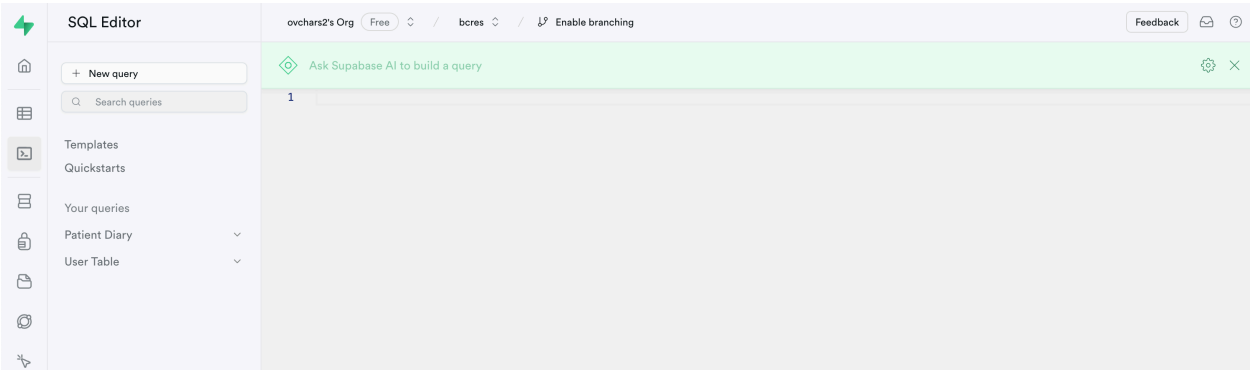


Рисунок 2.8 – Консольне введення SQL-запитів в Supabase

Supabase надає інтуїтивно зрозумілі інструменти візуалізації бази даних, які дозволяють користувачам легко переглядати та керувати структурою своїх даних через графічний інтерфейс користувача. Ці інструменти включають можливості для відображення схем, таблиць, відносин між таблицями та індексів, забезпечуючи розробникам глибше розуміння структури їх бази даних. Такий підхід спрощує процес проєктування, розвитку та оптимізації бази даних, роблячи його більш доступним і ефективним.

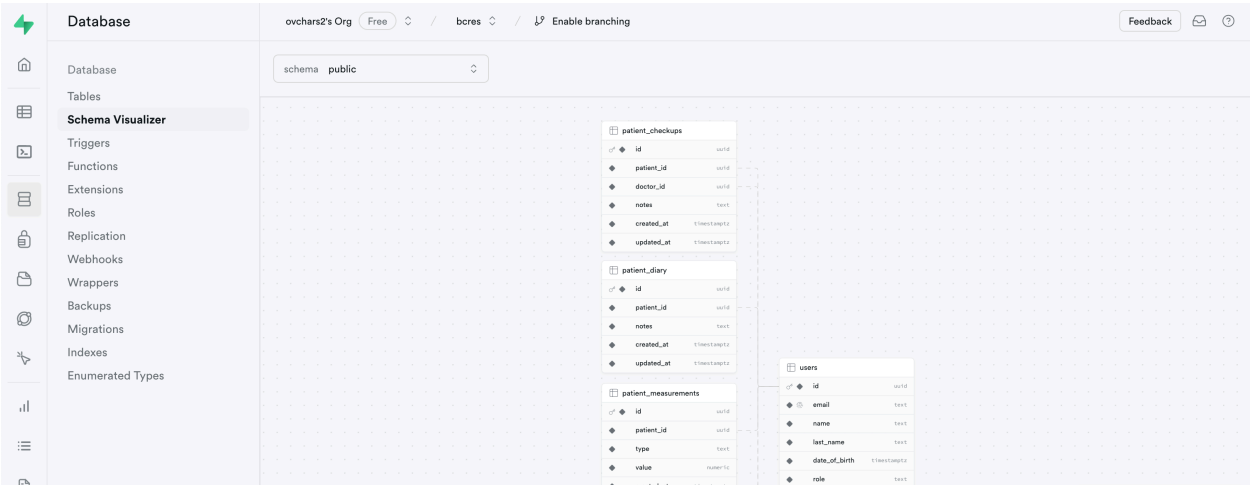


Рисунок 2.9 – Візуалізація зв’язків баз даних в Supabase

Supabase Edge Functions представляють собою потужний інструмент у екосистемі Supabase, який дозволяє створювати високопродуктивні веб-застосунки з низькою затримкою, розподіляючи логіку застосунку ближче до кінцевих користувачів, що забезпечує швидкий відгук застосунків та кращий користувацький досвід.

Основою Edge Functions є їх здатність виконувати код на мові JavaScript або TypeScript на периферійних серверах, розташованих у різних частинах світу. Це означає, що замість того, щоб весь серверний код виконувався в одному центральному місці, він може бути автоматично виконаний у місці, найближчому до користувача, мінімізуючи затримки та покращуючи загальну продуктивність застосунків.

Edge Functions ідеально підходять для різноманітних задач, від HTTP-запитів до складної логіки, такої як аутентифікація користувачів, персоналізація контенту та обробка форм. Це робить їх надзвичайно універсальними і дозволяє розробникам використовувати майже необмежені можливості для оптимізації.

Завдяки простоті в розгортанні та масштабуванні, Edge Functions відкривають двері до нових підходів у розробці, дозволяючи розробникам сконцентруватися на створенні високоякісного користувацького досвіду замість управління інфраструктурою. Ця модель розгортання коду на краю мережі не тільки забезпечує високу продуктивність та низьку затримку, але й знижує витрати та складність управління серверами, роблячи розробку застосунків швидшою, простішою та ефективнішою.

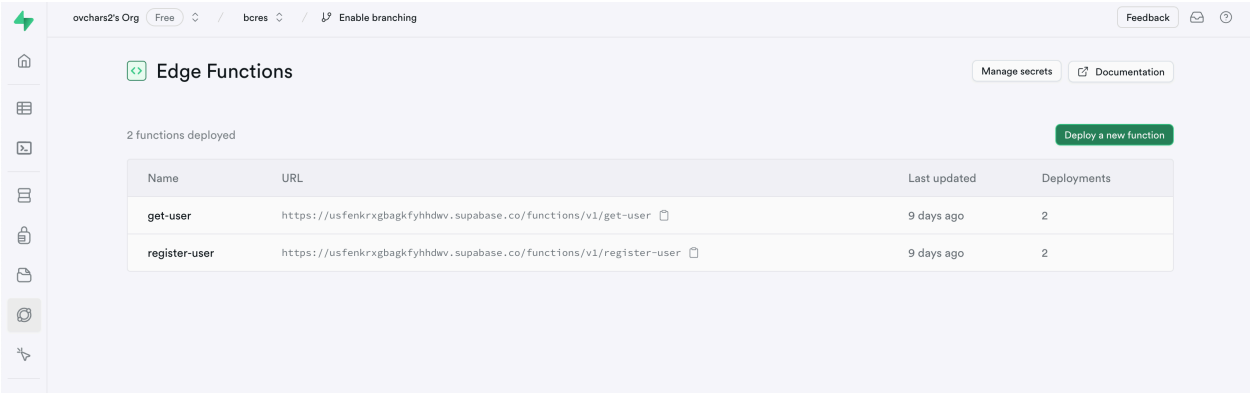


Рисунок 2.10 – Інтерфейс моніторингу Edge Functions в Supabase

Локальна розробка з Supabase пропонує розробникам гнучкість і контроль над їхніми проектами, дозволяючи роботу з Supabase на своєму локальному комп'ютері. Ця можливість сприяє більш швидкій реалізації

логіки, детальному тестуванню без необхідності постійного з'єднання з хмарним сервісом. Локальна розробка дозволяє розробникам використовувати всі основні функції Supabase, включаючи базу даних, аутентифікацію, оновлення в режимі реального часу та сховище в середовищі, яке повністю відтворює хмарне середовище Supabase з підтримкою можливості оновлення готових частин бізнес-логіки системи на хмарному середовищі виконанням однієї команди.

Використання локального середовища Supabase дозволяє налаштовувати застосунки в ізольованому середовищі, що забезпечує високий рівень безпеки та конфіденційності. Це особливо важливо при роботі з чутливими даними або при розробці функцій, що вимагають детального тестування перед розгортанням у хмарне середовище.

Локальне середовище розробки також сприяє співпраці в команді, оскільки розробники можуть легко поділитися своїм кодом і базами даних з колегами без необхідності надавати доступ до основного хмарного середовища. Це дозволяє командам ефективніше працювати над складними проектами, одночасно забезпечуючи цілісність даних.

Для налаштування локального середовища розробки з Supabase, розробники можуть скористатися офіційними інструментами та документацією, які надають інструкції щодо встановлення, конфігурації та використання Supabase локально [39]. Це включає в себе налаштування бази даних PostgreSQL, конфігурацію сервісів аутентифікації та авторизації, а також інтеграцію інших сервісів Supabase, таких як реальний час та сховище.

Крім того, локальна розробка сприяє глибшому розумінню роботи Supabase та PostgreSQL, оскільки розробники мають повний доступ до всіх аспектів своєї системи. Це може допомогти в ідентифікації та вирішенні потенційних проблем до того, як застосунок буде розгорнутий у хмарному середовищі.

2.3 Автоматизоване передавання показників пацієнтів для аналізу

Автоматизація передавання некритичних показань пацієнтів відіграє важливу роль у трансформації медичної індустрії, впливаючи на якість та ефективність надання медичних послуг. Цей процес забезпечує безперервний збір даних про стан здоров'я пацієнтів через використання цифрових технологій, що дозволяє медичному персоналу відстежувати показники в реальному часі. Це, в свою чергу, сприяє ранньому виявленню потенційних проблем, мінімізує необхідність у частих візитах до лікаря та дозволяє здійснювати своєчасне втручання, що може покращити результати лікування та знизити витрати на медичне обслуговування. Така автоматизація також сприяє підвищенню задоволеності пацієнтів, оскільки надає їм більше контролю над власним здоров'ям та сприяє розвитку персоналізованого підходу до лікування, заснованого на точних та актуальних даних.

Інтеграція автоматизованих систем передавання даних з електронними медичними записами та системами управління медичними закладами може значно покращити операційну ефективність, забезпечивши швидкий доступ до актуальної інформації про пацієнтів, оптимізувати розподіл ресурсів та поліпшити координацію догляду за пацієнтами. Автоматизація передавання некритичних показань пацієнтів має потенціал не тільки радикально змінити підходи до моніторингу та управління станом здоров'я на індивідуальному рівні, але й сприяти загальному розвитку медичної індустрії через покращення якості догляду, ефективності медичних досліджень та оптимізації медичних процесів.

Релевантними для аналізу та передбачення наявності раку молочної залози (див. 1.2) є наступні показники пацієнта:

- вік пацієнта;
- індекс маси тіла;
- рівень глюкози в крові;
- концентрація резистину в крові.

Вік пацієнта зазвичай визначається на основі дати народження, яку пацієнт може легко ввести в медичну документацію або мобільний застосунок без необхідності втручання медичного персоналу.

Індекс маси тіла (ІМТ), який обчислюється як відношення маси тіла пацієнта в кілограмах до квадрату його зросту в метрах, також може бути самостійно розрахований та внесений пацієнтом за умови наявності точних даних про масу тіла та зріст. Це можливо за допомогою домашніх ваг та рулетки або за допомогою спеціалізованих пристроїв, що автоматично синхронізують ці дані з медичними застосунками.

Рівень глюкози в крові може бути виміряний пацієнтом вдома за допомогою портативних глюкометрів, які дозволяють виконувати точні вимірювання та передавати дані безпосередньо в мобільні застосунки або електронні медичні записи, забезпечуючи зручний моніторинг.

Однак, визначення концентрації резистину в крові, який асоціюється з інсулінорезистентністю та метаболічними порушеннями, потребує більш складних лабораторних аналізів, що не можуть бути виконані пацієнтом самостійно. Такі аналізи вимагають забору крові та її подальшого дослідження в спеціалізованих лабораторіях з використанням високоточного обладнання, що забезпечує точне кількісне визначення рівнів резистину.

Таким чином, комплексний підхід до збору медичних даних, що поєднує самостійне введення інформації пацієнтами з клінічними аналізами, що виконуються медичними фахівцями, дозволяє отримати повну картину стану здоров'я пацієнта та забезпечує важливу основу для ефективного моніторингу, профілактики та лікування раку молочної залози.

2.4 Використання глюкометрів з можливістю передачі вимірювань через Bluetooth

Глюкометри, що передають дані за допомогою Bluetooth, забезпечують безперервний, точний і автоматизований збір даних про рівень глюкози, що

мінімізує людські помилки при введенні даних та знижує навантаження на пацієнтів, пов'язане з необхідністю регулярного введення показників вручну.

Автоматизована передача даних через Bluetooth у режимі реального часу не тільки спрощує процес збору даних, але й значно підвищує їхню надійність, оскільки дозволяє медичним фахівцям отримувати актуальну інформацію без затримок і перерв. Це особливо важливо для досліджень, де динаміка зміни рівня глюкози може надати інформацію про ризик розвитку певних захворювань, включаючи рак молочної залози.

Такий підхід сприяє ранньому виявленню потенційних проблем зі здоров'ям, підвищенню ефективності медичних процедур та розвитку новітніх діагностичних методик, заснованих на точних і об'єктивних даних, тим самим сприяючи загальному прогресу в медичній індустрії та покращенню якості життя пацієнтів.

Крім того, інтеграція даних з глюкометрів з електронними медичними системами та застосунками для моніторингу здоров'я дозволяє створювати комплексні бази даних, що можуть бути використані для глибокого аналізу, ідентифікації закономірностей та розробки персоналізованих методів лікування та профілактики.

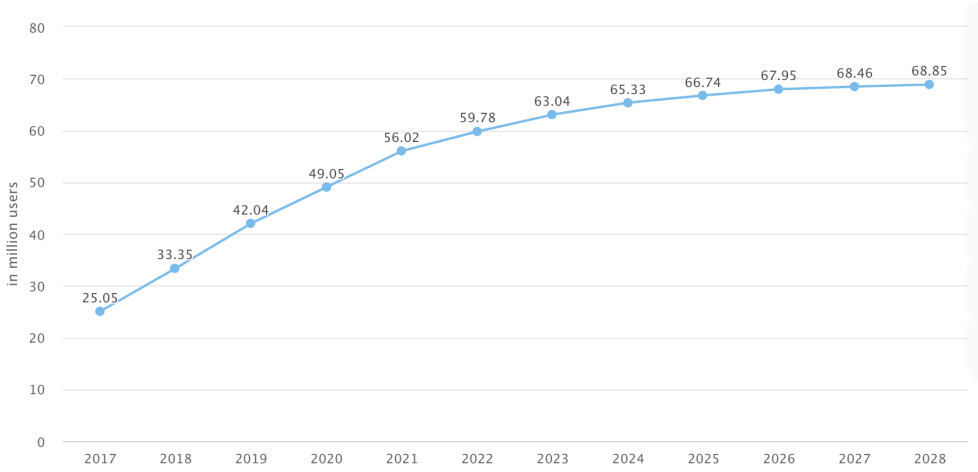


Рисунок 2.11 – Кількість користувачів розумних глюкометрів в світі [42]

Розумні глюкометри користуються значним попитом (див. рис. 2.11), оскільки вони спрощують процес моніторингу рівня глюкози в крові. Ці

пристрої автоматично передають вимірювання через Bluetooth на смартфон або інший мобільний пристрій, дозволяючи користувачам легко відстежувати свої показники та аналізувати тенденції. Інтеграція з мобільними застосунками також надає додаткові функції, такі як нагадування про вимірювання та поділ даних з лікарями, що надає більше можливостей контролю за здоров’ям.

Одні з найбільш популярних моделей розумних глюкометрів є Accu-Chek Instant та Contour Plus One.

Система Accu-Chek Instant забезпечує простий та зрозумілий досвід моніторингу рівня глюкози, забезпечуючи можливість простого самостійного моніторингу глюкози в крові. Система характеризується простотою у використанні: індикатор цільового діапазону візуально допомагає пацієнтам правильно інтерпретувати показники рівня глюкози в крові.

Рисунок 2.12 – Глюкометр Accu-Chek Instant

Функція простого підключення через Bluetooth забезпечує безперебійну передачу результатів вимірювань рівня глюкози до мобільних застосунків, сприяючи ефективнішому управлінню діабетом. Система відповідає міжнародним стандартам точності, зазначеним у ISO 15197:2013 / EN ISO 15197:2015 [43].

Широка зона дозування дозволяє наносити краплю крові на будь-яку частину краю смужки. Керований порт для смужки забезпечує легке та зручне введення смужки в глюкометр. Результати легко читаються на великому дисплеї з підсвіткою, що забезпечує зручність користування. Система також містить пристрій для викидання використаних тест-смужок, що дозволяє швидко та гігієнічно усунути використані смужки.

Основні характеристики системи Accu-Chek Instant [44]:

- Час вимірювання: приблизно 4 секунди (залежно від концентрації глюкози в крові);
- робоча температура від +10 до +40 °C (від +50 до +104 °F);
- робоча відносна вологість від 10 до 85 %;
- обсяг вибірки 0,6 мкл (мін. розмір зразка);
- діапазон гематокриту 10-65%;
- незалежність від висоти до 3094 метрів;
- автоматичне кодування;
- ємність пам'яті – щонайменше 720 результатів рівня глюкози в крові та щонайменше 30 контрольних результатів із зазначенням часу та дати; і середні значення за 7, 14, 30 і 90 днів;
- розміри: 77,1 × 48,6 × 15,3 мм (ДШВ);
- вага: 49 г (з елементами живлення);
- елементи живлення – 3В літієві батареї типу CR2032, 2шт;
- наявність Bluetooth з низьким енергоспоживанням, що працює в діапазоні частот від 2402 МГц до 2480 МГц з максимальною потужністю переданого сигналу 0 дБм (1 мВт) та інтеграцією Bluetooth Glucose Profile.

Глюкометр Contour Plus One забезпечує безперебійне підключення до смартфонів для запису всіх показників рівня глюкози в крові. Функція smartLIGHT дозволяє швидше та легше інтерпретувати показники рівня глюкози в крові за допомогою кольорових індикаторів, які чітко визначають,

чи є показник вище, у межах чи нижче цільового діапазону. Система Contour Plus One перевищила мінімальні вимоги до точності стандарту ISO 15197:2013 (EN ISO 15197:2015).

Рисунок 2.13 – Глюкометр Contour Plus One

Характеристики глюкометра Contour Plus One:

- час аналізу – 5 секунд;
- діапазон гематокриту 0 – 70%;
- діапазон результатів 0.6 – 33.3 ммоль/л;
- метод виміру – електрохімічний;
- об'єм зразка крові 0,6 мл;
- відносна вологість 10 – 93%;
- 800 результатів тестування крові з датою та часом;
- живлення за допомогою двох батарей CR-2032;
- робоча температура 5-45 °C;
- вага – 36 г;
- габаритні розміри – 97 х 28 х 14,9 мм.

Враховуючи наявність інтеграції з стандартизованим Bluetooth Glucose Profile, детальний опис інтерфейсу комунікації якого описано в наступному розділі, в обох моделях глюкометрів, та схожі характеристики, обидві моделі повинні інтегруватись в систему моніторингу без проблем, в основному, завдяки індустріальному стандарту комунікації більшості доступних глокометрів на ринку. Враховуючи популярність в Україні та зручність у використанні глюкометра Accu-Chek Instant, його було обрано як зразок розумного глюкометра для розробки комунікації з мобільним застосунком.

Висновки до розділу 2

Flutter був обраний для розробки мобільного застосунку завдяки його здатності створювати високопродуктивні крос-платформові застосунки з багатим користувацьким інтерфейсом із єдиної кодової бази. Flutter надає потужні інструменти для створення гнучкого дизайну, що важливо для створення інтуїтивно зрозумілого та привабливого користувацького досвіду.

Вибір Supabase як платформи для розробки бекенду обумовлений її широкими можливостями щодо швидкої розробки, масштабованості та легкості інтеграції з різними сервісами. Як повнофункціональна альтернатива Firebase, але з відкритим кодом та підтримкою SQL через PostgreSQL, Supabase пропонує розширені функції управління базами даних, автентифікації користувачів, зберігання файлів та миттєвих оновлень, що є вирішальним для ефективного збору та обробки даних у медичних застосунках. Ця платформа дозволяє розробникам швидко реалізовувати надійні та безпечні бекенд-системи, забезпечуючи високу продуктивність та доступність застосунків.

При виборі між моделями глюкометрів Contour Plus One та Accu-Chek Instant для тестування комунікації, перевага була надана Accu-Chek Instant. Accu-Chek Instant вирізняється більшою популярністю, простотою використання, зокрема завдяки інтуїтивному індикатору цільового діапазону та можливості легкого та гігієнічного видалення використаних тест-смужок. Ці особливості роблять можливою інтеграцію з медичними застосунками, спрямованими на моніторинг рівня глюкози в крові.

Інтегрований підхід, що об'єднує переваги Flutter для розробки мобільного застосунку, Supabase для реалізації бекенду та Accu-Chek Instant як оптимального пристрою для вимірювання глюкози, забезпечує міцну основу для створення застосунку. Враховуючи ці фактори, можна зробити висновок, що обраний технологічний стек і модель глюкометра відповідають вимогам до застосунку.

3 ІНТЕРФЕЙС GLUCOSE PROFILE ДЛЯ КОМУНІКАЦІЇ З ГЛЮКОМЕТРАМИ ТА ІНСТРУМЕНТИ ДЛЯ ІНТЕГРАЦІЇ ГЛЮКОМЕТРІВ В МОБІЛЬНІ ЗАСТОСУНКИ

Bluetooth Glucose Profile є частиною стандартів Bluetooth Health Device Profile (HDP), розроблених для бездротової передачі медичних даних між різними пристроями та застосунками. Цей профіль спеціалізується на передачі даних вимірювань рівня глюкози в крові від персональних глюкометрів до мобільних телефонів, планшетів, медичних записів або інших систем управління даними про здоров'я. Використання Bluetooth Glucose Profile уможливлює інтеграцію даних про глюкозу в ширші системи охорони здоров'я, забезпечуючи пацієнтам, лікарям та дослідникам доступ до точної та актуальної інформації, що може покращити управління діабетом та сприяти більш ефективному лікуванню.

Bluetooth Glucose Profile включає в себе декілька ключових аспектів, які забезпечують надійність, точність та безпеку передачі даних [45]:

- вимоги до кодування даних, які гарантують, що інформація про рівень глюкози передається без помилок та спотворень Це досягається за допомогою стандартизованих протоколів передачі даних, які визначають формати повідомлень та процедури їх обміну;
- профіль враховує необхідність захисту конфіденційної медичної інформації, впроваджуючи механізми шифрування та аутентифікації для запобігання несанкціонованому доступу до даних.

Окрім технічних аспектів передачі даних, Bluetooth Glucose Profile також включає в себе рекомендації щодо інтерпретації отриманих даних. Це означає, що профіль не тільки фокусується на технологічній стороні збору та передачі інформації, але й надає підтримку в аналізі та використанні даних для моніторингу стану здоров'я. Такий підхід сприяє інтеграції даних про глюкозу

в комплексні програми управління діабетом, дозволяючи автоматизувати деякі аспекти моніторингу та надаючи пацієнтам та лікарям інструменти для більш ефективного реагування на зміни у рівні глюкози.

Цей профіль демонструє, як інноваційні технології можуть сприяти більш тісній взаємодії між пацієнтами та медичними фахівцями, сприяючи розробці індивідуальних лікувальних стратегій на основі детального аналізу даних про глюкозу. Крім того, інтеграція даних з глюкометрів у ширші системи здоров'я сприяє збору великих обсягів медичних даних, які можуть бути використані для наукових досліджень, спрямованих на вдосконалення методів діагностики, моніторингу та лікування різноманітних захворювань.

3.1 Інтерфейс Bluetooth Glucose Profile 1.0.1

Профіль визначає дві ролі: датчик глюкози та колектор. Датчик глюкози - це пристрій, який вимірює концентрацію рівня глюкози, а колектор – це пристрій, який отримує вимірювання та інші пов’язані дані з датчика глюкози. Як датчик глюкози, так і Колектор має бути сервером Generic Attribute Profile (GATT), що є стандартом для обміну даними для Bluetooth Low Energy [46].

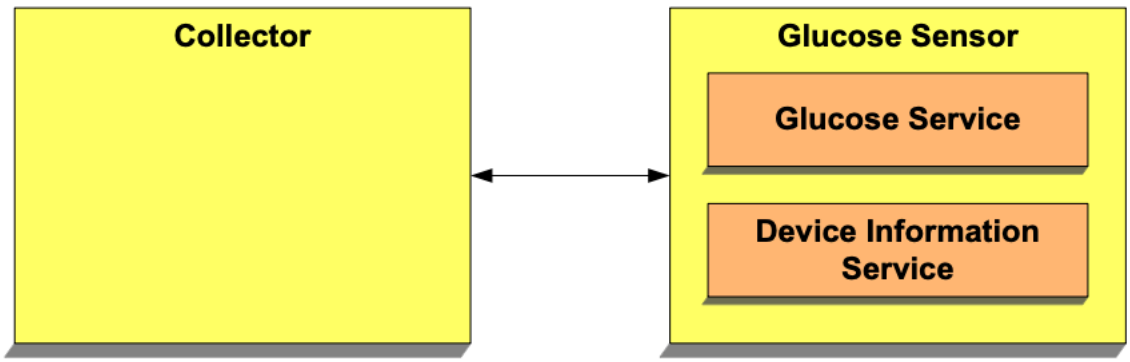


Рисунок 3.1 – Структура зв’язку Датчик-Колектор в Glucose Profile

Датчик глюкози, в свою чергу, відповідає за створення двох окремих сервісів для збору даних про рівень глюкози в крові та загальну інформацію про пристрій – Glucose Service та Device Information Service (див. рис. 3.1).

Датчик глюкози та Колектор мають використовувати периферійну та центральну ролі Generic Access Profile (GAP) [47] відповідно.

Glucose Profile повинен працювати виключно з Low-Energy (LE) передачею даних.

Перебуваючи в режимі виявлення GAP для початкового підключення до колектора, датчик глюкози повинен містити UUID служби глюкози в полі типу AD UUID служби в рекламних даних. Це покращує взаємодію з користувачем, оскільки датчик глюкози може бути визначений колектором до встановлення з’єднання.

Для покращення взаємодії з користувачем датчик глюкози повинен містити локальну назву (що містить повне або скорочене значення характеристики назви пристрою) у своїх рекламних даних або даних відповіді сканування.

Датчик глюкози може підтримувати властивість запису для характеристики назви пристрою, щоб дозволити Колектору записати назву пристрою в датчик глюкози.

Для покращення взаємодії з користувачем датчик глюкози, який підтримує кілька зв’язків, може включати тип AD загальнодоступної цільової адреси або випадковий цільовий тип AD у своїх даних відповіді сканування. Якщо датчик глюкози не підтримує кілька зв’язків, датчик глюкози не повинен використовувати тип цільової адреси AD.

Значення біта Multiple Bond Supported характеристики функції глюкози має бути встановлено відповідно до функціональних можливостей датчика глюкози, щоб колектор міг визначити, чи датчик глюкози не підтримує тип AD цільової адреси (тобто якщо біт Multiple Bond Supported дорівнює 0) або потенційно підтримує тип AD Target Address (тобто якщо біт Multiple Bond Supported дорівнює 1).

Додаткові вимоги до Device Information Service датчика глюкози передбачають наявність унікального номеру виробника та моделі глюкометра, та наявність ідентифікатора системи.

Наведені вище характеристики можуть бути прочитані Колектором для подальшого використання в екосистемі – за замовчуванням, Колектор не зобов’язаний читати дані з сервісу Device Information датчика глюкози.

Необхідні та опціональні вимоги до підтримки субпроцедур GATT Колектором наведено в таблиці 3.1

Таблиця 3.1 Вимоги до підтримки субпроцедур GATT Колектором

Вимога	Підтримка колектором
Service Dicsovery	
Glucose Service Discovery	Обов’язкова
Device Information Service Discovery	Опціональна
Characteristics Discovery	
Glucose Service Characteristics Discovery	Обов’язкова
Device Information Service Characteristics Discovery	Опціональна
Glucose Measurement	Обов’язкова
Glucose Measurement Context	Обов’язкова
Glucose Feature	Обов’язкова
Record Access Control Point	Обов’язкова

Колектор повинен мати можливість виявити Glucose Service та його характеристики, але не зобов’язаний виявляти сервіс та характеристики, пов’язані з Device Information Service.

Підтримка характеристик Glucose Measurement та контексту й додаткової інформації щодо вимірів глюкози обов’язкова до імплементації. Перед виконанням будь-якої процедури Колектор повинен налаштувати характеристику вимірювання рівня глюкози для сповіщень (тобто через дескриптор конфігурації характеристик клієнта). Коли Колектору потрібні дані про глюкозу, він повинен дотримуватися відповідних процедур.

Колектор повинен визначити вміст характеристичної структури вимірювання рівня глюкози на основі вмісту поля «Flags». Це дозволяє

Колектору визначити одиницю вимірювання поля «Glucose Concentration» та наявність чи відсутність додаткових полів, визначених характеристикою. Колектор повинен мати можливість приймати обидві одиниці поля концентрації глюкози (тобто в базових одиницях кг/л і моль/л).

Колектор повинен прочитати характеристику Glucose Feature, щоб визначити підтримувані функції датчика глюкози, щоб інтерпретувати біти в сповіщенні про стан характеристики Glucose Measurement. Наприклад, якщо функція «Low Battery Detection During Measurement» не підтримується, тоді біт «Device Battery Low» не матиме значення. З іншого боку, якщо функція виявлення низького заряду батареї під час вимірювання підтримується, тоді біт пристрою «Device Battery Low» вказуватиме на те, що заряд батареї датчика був виявлений низьким під час даного вимірювання, чи ні.

Якщо датчик глюкози підтримує індикацію характеристики Glucose Feature, Колектор може налаштувати цю характеристику для індикації. Коли колектор отримує індикацію функції Glucose Feature, Колектор повинен використовувати вказане значення для повторного визначення підтримуваних функцій. Крім того, Колектор може зчитувати характеристику Glucose Feature кожного разу після підключення до датчика глюкози.

Якщо біт «Low Battery Detection During Measurement» встановлено на 0 (False), колектор повинен ігнорувати біт «Device Battery Low» в полі «Sensor Status Annunciation».

Якщо біт «Sensor Malfunction Detection Supported» встановлено на 0 (False), колектор повинен ігнорувати біт «Sensor Malfunction or Faulting» поля «Sensor Status Annunciation».

Якщо біт «Sensor Sample Size Supported» встановлено на 0 (False), колектор повинен ігнорувати біт «Sample Size for Blood or Control Solution Insufficient».

Якщо біт «Sensor Strip Insertion Error Detection Supported» встановлений на 0 (False), колектор повинен ігнорувати біт «Strip Insertion Error».

Якщо біт «Sensor Strip Type Error Detection Supported» встановлений на 0 (False), колектор повинен ігнорувати біт «Strip Incorrect For Device».

Якщо біт «Sensor Result High-Low Detection Supported» встановлено на 0 (False), колектор ігнорує біти «Sensor Result Higher Than the Device Can Process» та «Sensor Result Lower Than the Device Can Process»

Якщо біт «Sensor Temperature High-Low Detection Supported» встановлено на 0 (False), колектор ігноруватиме біт «Sensor Temperature Too High for Valid Test/Result» та біт «Sensor Temperature Too High for Valid Test/Result».

Якщо біт «Sensor Read Interrupt Detection Supported» встановлено на 0 (False), колектор ігноруватиме біт «Sensor Read Interrupted Because Strip Was Pulled Too Soon».

Якщо біт «General Device Fault Supported» встановлено на 0 (False), колектор ігнорує біт «General Device Fault Has Occurred».

Якщо біт «Time Fault Supported» встановлено на 0 (False), колектор має ігнорувати біт «Time Fault Has Excursed in the Sensor» та «Time May Be Inaccurate»

Якщо біт «Multiple Bond Supported» встановлений на 0 (False), Колектор може визначити, що датчик глюкози підтримує лише одиничний зв’язок. Інакше Колектор може визначити, що датчик підтримує множинні зв’язки.

Якщо Колектор зчитує або отримує характеристику Glucose Feature із встановленими бітами, позначеними як «Зарезервовано для майбутнього використання» (RFU), Колектор має ігнорувати ці біти та продовжувати працювати в звичайному режимі.

В контексті Glucose Service, записи складаються з характеристики Glucose Measurement, та можуть додатково мати значення Glucose Measurement Context характеристики. Зміст повідомлень має визначатись Колектором.

Таким чином, використання Glucose Profile дозволяє розробити систему комунікації з датчиками рівня глюкози в крові та автоматизувати збір даних.

3.2 Використання Flutter Method Channel для комунікації з нативними сервісами Android та iOS

Flutter використовує гнучку систему, яка дозволяє викликати специфічні для платформи API мовою, яка працює безпосередньо з цими API:

- Kotlin або Java на Android
- Swift або Objective-C на iOS
- C++ у Windows
- Objective-C на macOS
- C на Linux

Вбудована підтримка API Flutter для конкретної платформи не покладається на генерацію коду, а скоріше на гнучкий стиль передачі повідомлень. Частина програми Flutter надсилає повідомлення своєму хосту, частині програми, що не написана мовою програмування Dart, через канал платформи (Method Channel).

Хост прослуховує Method Channel та отримує повідомлення. Потім він викликає будь-яку кількість специфічних для платформи API, використовуючи рідну для платформи мову програмування, і надсилає відповідь клієнту, частині програми Flutter.

На стороні клієнта MethodChannel дозволяє надсилати повідомлення, які відповідають викликам методів. На стороні платформи MethodChannel на Android (MethodChannelAndroid) і FlutterMethodChannel на iOS (MethodChanneliOS) дозволяють отримувати виклики методів і надсилати результат. Ці класи дозволяють розробити плагін платформи без необхідності написання великої кількості «шаблонного» коду.

Повідомлення передаються між клієнтом (UI) і хостом (платформою) за допомогою каналів платформи, як показано на рисунку 3.2.

Повідомлення та відповіді передаються асинхронно, щоб інтерфейс користувача не страждав від затримок пов’язаних з комунікацією.

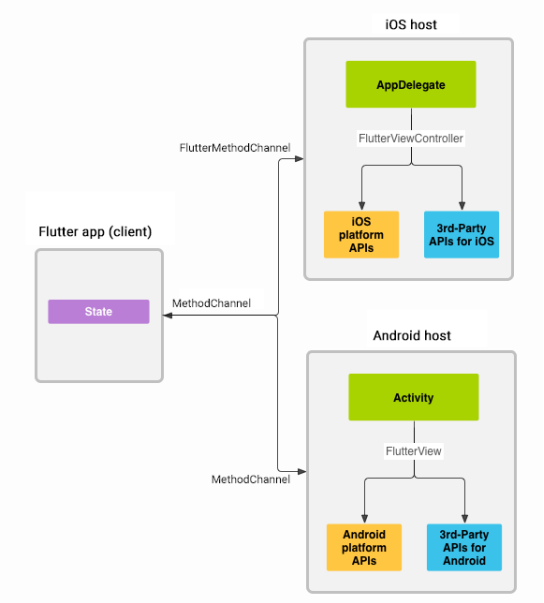


Рисунок 3.2 – Схема передачі повідомлень Method Channel

Стандартні канали платформи використовують стандартний кодек повідомлень, який підтримує ефективну двійкову серіалізацію простих значень, подібних до JSON, таких як логічні значення, числа, рядки, буфери байтів, а також їх списки та словники. Серіалізація та десеріалізація цих значень до та з повідомлень відбувається автоматично, під час надсилання та отримання значень.

Перевага використання стандартного кодека повідомлень полягає в тому, що він підтримує надсилання простих JSON-подібних значень, таких як:

– null;
– boolean;
– числа (цілі та з плаваючою комою);
– String(кодування UTF-8);
– списки елементів примітивних типів;
– Map (словники з String ключами);
– двійкові дані (масиви байтів).

Стандартний кодек повідомлень забезпечує ефективну двійкову серіалізацію, використовуючи найменшу кількість пам’яті та обчислювальних ресурсів і забезпечуючи швидке виконання. Це налаштування допомагає підтримувати оперативність інтерфейсу користувача під час надсилання та отримання викликів методів. Ці двійкові кодеки забезпечують правильний переклад отриманих даних на нативній стороні та ефективне використання у нативних реалізаціях.

Таблиця 3.2. Конвертація типів даних у MethodChannel

Dart(Flutter)	Kotlin(Android)	Swift(iOS)
null	null	nil
bool	Boolean	NSNumber(value: Bool)
int	Int	NSNumber(value: Int32)
int, якщо 32 біти недостатньо	Long	NSNumber(value: Int)
double	Double	NSNumber(value: Double)
String	String	String
Uint8List	ByteArray	FlutterStandardTypedData(bytes: Data)
Int32List	IntArray	FlutterStandardTypedData(int32: Data)
Int64List	LongArray	FlutterStandardTypedData(int64: Data)
Float32List	FloatArray	FlutterStandardTypedData(float32: Data)
Float64List	DoubleArray	FlutterStandardTypedData(float64: Data)
List	List	Array
Map	HashMap	Dictionary

Незважаючи на те, що MethodChannel дуже корисний, важливо знати його обмеження та способи їх подолання.

Стандартний кодек повідомлень може кодувати та декодувати лише певні типи даних. Таким чином, MethodChannel підтримує обмежений набір даних. Щоб подолати це, можливо серіалізувати власні дані у форматі, який розуміє MethodChannel, наприклад JSON, а потім десеріалізувати їх з іншого боку.

Якщо нативний код видає помилку, Flutter не може її зафіксувати. Потрібно обробляти всі можливі помилки на стороні платформи.

MethodChannel підтримує надсилання та отримання одного повідомлення. Він не підтримує потокові дані. У таких випадках можна використовувати EventChannel або StreamChannel.

Якщо два виклики методів надсилаються паралельно, один повинен чекати, поки інший завершиться, оскільки Dart є однопотоковим.

Для MethodChannel потрібен специфічний для платформи код для iOS і Android. Це може збільшити кількість коду, який потрібно написати та підтримувати.

Наскільки це можливо, використання плагінів Flutter, які вже обробляють Platform Channel, може бути більш ефективним і зменшити робоче навантаження.

3.3 Бібліотека flutter_blue_plus для Bluetooth комунікації в Flutter

FlutterBluePlus (flutter_blue_plus) — це плагін Flutter, який спрощує зв’язок Bluetooth Low Energy (BLE). Він ґрунтується на оригінальному пакеті flutter_blue [48] і вдосконалює його завдяки покращеній стабільності, виправленню помилок і новим функціям. Цей пакет дозволяє вам виконувати такі завдання, як сканування пристроїв BLE поблизу, підключення до них, читання та запис характеристик і підписка на сповіщення.

FlutterBluePlus підтримує майже всі функції на всіх підтримуваних платформах: iOS, macOS, Android. FlutterBluePlus був написаний як простий, надійний і зрозумілий. FlutterBluePlus не має жодних залежностей, окрім самих Flutter, Android та iOS. Кожна помилка, яку повертає нативна платформа, повертається на клієнтську частину, де це необхідно.

Потоки, які повертає FlutterBluePlus, ніколи не видають жодних помилок і ніколи не закриваються. Немає необхідності обробляти onError або onDone для stream.listen. Єдиним винятком є FlutterBluePlus.scanResults, який слід обробити додаванням параметру onError.

flutter_blue_plus сумісний лише з версіями Android SDK вище за 21, тому слід змінити це в файлі конфігурації android/app/build.gradle.

Доступні методи бібліотеки FlutterBluePlus з різних частин API, як власних методів так і методів Bluetooth, наведено в таблицях 3.3 – 3.5.

Таблиця 3.3 – Методи FlutterBluePlus API

Метод	Android	iOS	Опис
setLogLevel	+	+	Налаштування рівня логування
isSupported	+	+	Перевіряє, чи пристрій підтримує Bluetooth
turnOn	+		Вмикає адаптер Bluetooth
adapterStateNow	+	+	Поточний стан Bluetooth адаптера
adapterState	+	+	Потік увімкнення та вимкнення адаптера Bluetooth
startScan	+	+	Початок сканування пристроїв BLE
stopScan	+	+	Зупиніть наявне сканування пристроїв BLE
onScanResults	+	+	Потік результатів сканування в реальному часі
scanResults	+	+	Потік результатів сканування в реальному часі або попередніх результатів
lastScanResults	+	+	Останні результати сканування
isScanning	+	+	Потік поточного стану сканування
isScanningNow	+	+	Перевірка виконування сканування в даний моемент
connectedDevices	+	+	Список пристроїв, підключених до програми
systemDevices	+	+	Список пристроїв, підключених до системи, навіть за допомогою інших програм
getPhySupport	+		Підтримувані кодування Bluetooth PHY

Таблиця 3.4 – Методи BluetoothDescriptor API

Метод	Android	iOS	Опис
uuid	+	+	uuid дескриптора
read	+	+	Отримує значення дескриптора
write	+	+	Записує значення дескриптора
onValueReceived	+	+	Потік значення дескриптора читає та записує
lastValue	+	+	Останнє значення дескриптора
lastValueStream	+	+	Потік onValueReceived + записи

FlutterBluePlus представляє всі методи бібліотеки, що доступні для виклику для налаштувань роботи з інструментами.

BluetoothCharacteristic API представляє характеристику Bluetooth GATT – базовий елемент даних, який використовується для

створення служби GATT, BluetoothGattService. Характеристика містить значення, а також додаткову інформацію та додаткові дескриптори GATT, BluetoothGattDescriptor.

Таблиця 3.5 – Методи BluetoothCharacteristicAPI

Метод	Android	iOS	Опис
uuid	+	+	Uuid характеристики
read	+	+	Отримує значення характеристики
write	+	+	Записує значення характеристики
setNotifyValue	+	+	Встановлює сповіщення або індикацію на характеристику
isNotifying	+	+	Перевіряє сповіщення або індикацію
onValueReceived	+	+	Потік оновлень значень характеристик, отриманих від пристрою
lastValue	+	+	Останнє значення характеристики
lastValueStream	+	+	Uuid характеристики

BluetoothDesrtiptor API представляє дескриптор Bluetooth GATT. Дескриптори GATT містять додаткову інформацію та атрибути характеристики GATT, BluetoothGattCharacteristic. Їх можна використовувати для опису ознак характеристики або для контролю певної поведінки характеристики.

Висновки до розділу 3

Розуміння Bluetooth Glucose Profile, володіння методами Flutter Method Channels та використання бібліотеки flutter_blue_plus є важливими складовими для розробки мобільного застосунку. Використання Bluetooth Glucose Profile дозволяє застосунку точно отримувати дані з глюкометрів, що підтримують Bluetooth. Flutter Method Channels відіграють ключову роль у забезпеченні двостороннього зв'язку між нативним кодом та кодом Dart у застосунку Flutter. Використання бібліотеки flutter_blue_plus надає потужний інструментарій для роботи з Bluetooth у Flutter-застосунках, спрощуючи процес розробки та забезпечуючи високу сумісність з різноманітними пристроями.

4 РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ МОНІТОРИНГУ

Проведений аналіз аналогічних рішень в сфері надання медичних послуг та контролю за здоров’ям, оцінку технологічних рішень для розробки сучасних клієнт-серверних систем, а також дослідження інтерфейсів комунікації та методів для організації комунікації з глюкометрами дає можливість продовжити розробку інтелектуальної системи моніторингу та діагностики раку молочної залози з урахуванням вимог до сучасних мобільних застосунків, та використовуючи новітні методи та технології розробки.

4.1 Архітектура та функції мобільного застосунку

Мобільний застосунок системи діагностики та моніторингу раку молочної залози повинен забезпечувати індивідуалізований підхід до кожного пацієнта, дозволяючи адаптувати рекомендації та плани лікування відповідно до особистих даних та результатів аналізів і обстежень. Окрім цього, в застосунку мають бути доступні інструменти, що дають змогу лікарям вносити та контролювати зміни стану пацієнтів.

Ключові особливості мобільного застосунку для діагностики раку молочної залози повинні включати [3]:

- інтерфейс для лікарів, в якому лікарі зможуть переглядати медичні дані пацієнтів, отримувати звіти про стан здоров'я, спостерігати за динамікою захворювання та надавати рекомендації пацієнтам;
- інтерфейс для пацієнтів, в якому пацієнти отримають можливість вести журнал свого здоров'я, отримувати відгуки від лікарів та консультуватися з ними онлайн;
- автоматизований збір даних, що виконується за рахунок синхронізації з зовнішніми медичними пристроями, які використовує пацієнт (ваги, глюкометр тощо) для автоматичного збору важливих даних про стан здоров'я, що може вказувати на розвиток або прогресування хвороби;

– можливість завантажувати результати нових медичних досліджень у базу даних застосунку для подальшого аналізу;
– використання розширених алгоритмів діагностики, що дозволить застосунку автоматично аналізувати зібрані медичні дані, виявляти патологічні зміни та інформувати про потенційні ризики для здоров'я, які потребують консультації з фахівцем.

Поєднання Flutter та Supabase для розробки застосунку забезпечить високу швидкість розробки, безпеку даних та легкість масштабування архітектури.

Рисунок 4.1 – Функції мобільного застосунку для лікарів та пацієнтів

Архітектура мобільного застосунку MVC (Model-View-Controller) дозволить розділити дані (Model), інтерфейс (View) та логіку управління (Controller), спрощуючи розробку та тестування застосунку. Flutter має достатню кількість вбудованих функцій для побудови архітектури MVC.

Деякі функції мобільного застосунку – аутентифікація, оновлення даних в режимі реального часу – можливо реалізувати за допомогою прямої комунікації Flutter-застосунку з сервісами Supabase.

Для контролю доступу до даних за рахунок визначення ролі користувача (пацієнт або лікар), управління та обміну інформацією між лікарями та пацієнтами можливе створення RESTful API з використанням Supabase Edge

Functions, що виступає в ролі middleware між даними з баз даних Supabase та користувачами мобільних застосунків. Такий підхід запобігає неавторизованому доступу до конфіденціальної інформації.

4.2 Структура мобільного застосунку

Для розробки та релізу застосунку використовується Flutter версії 3.16, що надає розробникам додаткові функції та зручність розробки за допомогою нових функцій Dart версії 3.

Для розробки багатомовності застосунку використовується бібліотека easy_localization для Flutter. Ця бібліотека дозволяє розробникам легко і ефективно інтегрувати локалізацію у свої застосунки, підтримуючи багатомовність. Використання easy_localization значно спрощує процес додавання, оновлення та управління мовними ресурсами.

Основною перевагою easy_localization є її здатність автоматизувати багато аспектів локалізації, таких як вибір мови в залежності від налаштувань системи користувача або вимог додатку. Бібліотека підтримує використання JSON та CSV файлів для зберігання перекладів, що робить їх читабельними та зручними для редагування як розробниками, так і фахівцями з локалізації.

Впровадження easy_localization у мобільний застосунок на Flutter починається з додавання залежності до файлу pubspec.yaml, після чого з’являється можливість налаштувати основні параметри локалізації, такі як підтримувані мови та шлях до файлів з перекладами. Наступним кроком є ініціалізація easy_localization у головному файлі застосунку, де вказуються вихідні параметри, такі як мова за замовчуванням та контекст локалізації.

Для управління конфігурацією середовища в застосунку використовується бібліотека flutter_dotenv. Її основне призначення полягає у забезпеченні безпечного та зручного механізму інтеграції змінних середовища, що дозволяє зберігати конфіденційні дані, такі як ключі API, конфігурації баз даних та інші чутливі параметри, поза вихідним кодом. Впровадження цієї бібліотеки в застосунок вимагає додавання залежності до

файлу pubspec.yaml, а потім створення та конфігураційного файлу .env, де зберігатимуться змінні середовища.

Впровадження бібліотеки go_router для навігації в застосунку вимагає ініціалізації маршрутизатора на ранньому етапі створення застосунку, з подальшим описом маршрутів вказанні конкретних відображень екранів, які викликатимуться на кожному із маршрутів. Ця бібліотека забезпечує глибоку інтеграцію з Flutter, підтримуючи такі функції, як відкладена загрузка (lazy loading), передачу параметрів між екранами та збереження стану в навігації.

Для збереження та передавання даних про стан застосунку в кожний момент часу буде використовуватись патерн проектування мобільних застосунків BLoC (Business Logic Component). Цей патерн було запропоновано компанією Google як такий, що має потенціал для масштабування, визначається структурованістю та легкістю тестування окремого функціоналу.

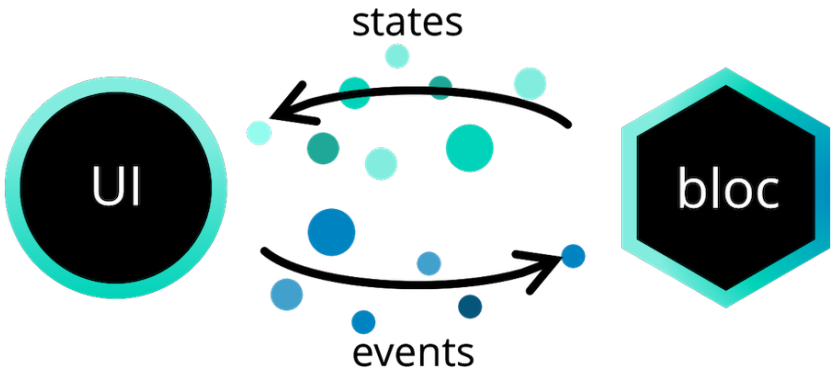


Рисунок 4.2 – Патерн BLoC

Структура патерну складається з двох асинхронних потоків, що поєднують інтерфейс користувача та бізнес-логіку застосунку. Потік подій (Events), які надходять до компоненту, надає можливість асинхронно реагувати на події різних типів – класифікувати типи подій та викликати окремі методи для реакції на них. Реакцією на події, в залежності від їх типів та інформації, що в них закладено, є надсилання у потік станів (States) нової інформації про стан компонента.

Коли віджети Flutter-застосунку, що підписані на оновлення стану компонента BLoC, отримують нову подію через асинхронний потік даних,

викликаються методи, що забезпечують відображення коректної інформації у інтерфейсі користувача.

4.3 Структура бази даних

Для збереження даних було використано сервіс Supabase Database, що дає змогу виконувати запити для збереження даних як безпосередньо з застосунку, так і з використанням API на базі Supabase Edge Functions. У кожному з запитів є можливість перевірки статусу аутентифікації та виконання власних запитів для перевірки наявності в користувача доступу до відповідних таблиць або конкретних рядків бази даних.

Рисунок 4.3 – Структура бази даних застосунку

Наведена на рисунку 4.3 структура бази даних передбачає наступні таблиці та зв’язки між ними:

– таблиця users зберігає основні дані про користувача – ім’я, дата народження, адреса електронної пошти, дата створення та оновлення запису в

таблиці, а також роль користувача в системі – лікар або пацієнт. На основі даних, що зберігаються в цій таблиці, мобільний застосунок обиратиме відповідний інтерфейс для відображення користувачу;

– таблиця patients_assigned зберігає зв’язки пацієнтів та лікарів, що дає змогу пацієнтам зв’язуватись з лікарем, а лікарям – переглядати дані пацієнтів;

– таблиця patient_measurements зберігає результати аналізів пацієнтів, що вносяться або пацієнтом власноруч, або лікарем;

– таблиця patient_diary зберігає записи щоденника здоров’я пацієнта, що створюються пацієнтом власноруч;

– таблиця patient_checkups зберігає дані проведених оглядів пацієнта.

Кожна з таблиць захищена від неавторизованого перегляду за допомогою Row Level Security. Пацієнти мають право переглядати власні записи, а також можуть мати доступ до загальних даних (імені та адреси електронної пошти) свого лікаря. Лікарі мають право переглядати всі дані пацієнтів, що перебувають в них на обліку.

4.4 Аутентифікація та авторизація користувачів

Ключовим компонентом для аутентифікації користувача повинен бути компонент AuthBloc. Його головна задача – слідкувати за оновленнями стану авторизації та надавати стани «Authenticated» або «Unauthenticated», які відображають, чи є користувач авторизованим.

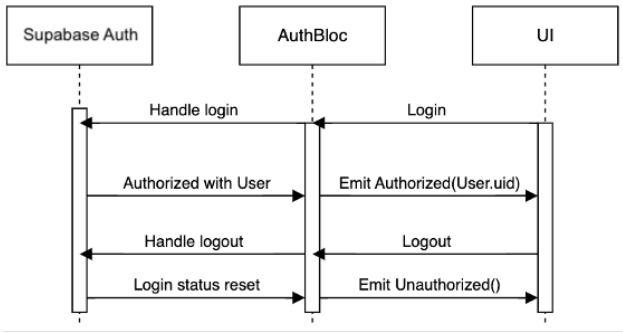


Рисунок 4.4 – Структура AuthBloc

Під час взаємодії з Supabase Auth, AuthBloc відстежує статус авторизації. Він перевіряє, чи користувач авторизований або неавторизований. За

допомогою методу onAuthStateChange() Supabase Authentication, AuthBloc підписується на зміни стану авторизації. Коли статус авторизації змінюється, AuthBloc оновлює свій стан відповідно до статусу аутентифікації користувача.

У випадку, коли користувач успішно пройшов аутентифікацію, мобільний застосунок повинен виконати первинну авторизацію користувача, визначивши його роль – пацієнт чи лікар.

Для проведення авторизації застосунок викликає Supabase Function з назвою get_user, що передає дані користувача, якщо вони доступні. У випадку відсутності даних про користувача функція надсилає відповідь з кодом помилки 404.

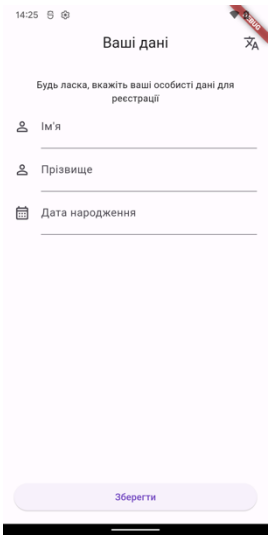


Рисунок 4.5 – Екран реєстрації користувача

Якщо мобільний застосунок отримує дані про користувача, користувач перенаправляється на екран головної сторінки інтерфейсу залежно від ролі користувача. У випадку відсутності даних про користувача в системі, застосунок показує користувачеві екран реєстрації, де ці дані мають бути введені та надіслані до бази даних за допомогою Supabase Function з назвою register_user. У випадку успішної реєстрації, користувача буде перенаправлено на головний інтерфейс залежно від його ролі.

Головний інтерфейс для пацієнта передбачає можливість перегляду останніх результатів вимірювань та їх оновлення – вручну для пацієнтів

доступне оновлення показників зросту та ваги. У випадку якщо пацієнт власноруч додає запис про концентрацію глюкози в крові, відкривається інтерфейс для автоматичного збору даних з глюкометра за допомогою Bluetooth (див. рис.4.8).

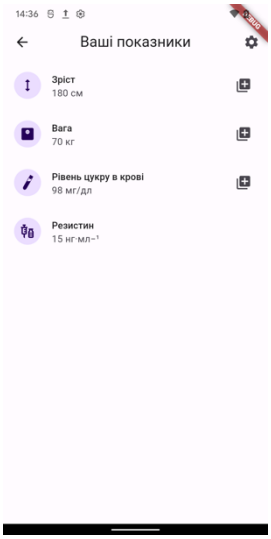


Рисунок 4.6 – Екран перегляду результатів вимірів користувача

Інтерфейс для лікарів має схожий до наведеного на рисунку 4.6 функціонал, що має використовуватись для внесення даних про стан пацієнта.

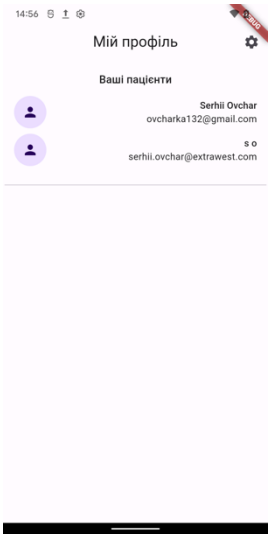


Рисунок 4.7 – Головна сторінка інтерфейсу для лікарів

У інтерфейсі для лікарів розблокована можливість ручного внесення актуальних даних про кожен з показників. Перехід до внесення даних

здійснюється з наведеного на рисунку 4.7 екрану, де лікар має можливість переглядати список його пацієнтів.

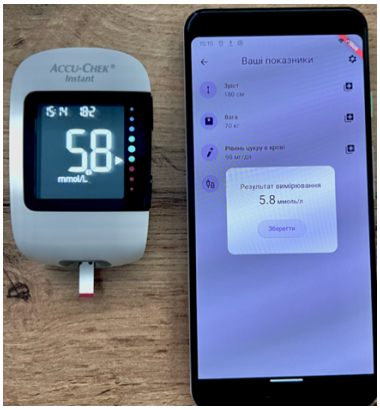


Рисунок 4.8 – Передача вимірювань рівня глюкози в крові

Для реалізації функціоналу задіяні компоненти PatientBloc, який за допомогою Supabase функції get-assigned-doctor отримує дані про основні дані пацієнта та його лікаря, а також PatientMeasurementsCubit, який отримує та надсилає дані про показники пацієнта для подальшого аналізу на основі даних, отриманих з бази даних за допомогою Supabase функції my-measurements.

Код мобільних компонентів та функцій Supabase наведений в додатку Б.

4.5 Висновки до розділу 4

В результаті виконання проектування архітектури застосунку, вибору бібліотек для розробки та власне процесу розробки програмного забезпечення було створено комплексну систему для моніторингу та діагностики захворювань, що використовує Supabase для авторизації, збереження даних та комунікації з бекенд частиною системи через API, що був створений з використанням Supabase Edge Functions. Система дає змогу пацієнтам актуалізувати дані вимірювань, що можливо провести в домашніх умовах власноруч або за допомогою спеціальних пристроїв. Лікарі, в свою чергу, матимуть можливість автоматично вносити дані кожного з вимірів кожного з їхніх пацієнтів. Завдяки використанню стандартизованих інтерфейсів комунікації та крос-платформової розробки, система буде доступною для великої кількості користувачів.

ВИСНОВКИ

В результаті роботи було створено інтелектуальну систему діагностики та моніторингу раку молочної залози, що дає змогу пацієнтам та лікарям автоматизувати збір даних, релевантних для подальшого аналізу.

Система, заснована на бекенд-платформі, що надає велику кількість сервісів та підтримує всі необхідні інструменти розробки, а також фреймворк розробки мобільних застосунків з великою кількістю активних розробників та відкритих бібліотек для організації комунікації, виявили себе як комбінація інструментів розробки, що має перспективи не лише в рамках проведеної роботи, а і для подальшої розробки масштабованих мобільних застосунків подібної архітектури. В поєднанні з стандартизованим інтерфейсом комунікації з мобільними датчиками, дана система буде доступною для великої кількості потенційних користувачів незалежно від операційної системи, виробників або моделей пристроїв, якими вони користуються.

Поставлені до роботи завдання було виконано – проведено огляд сучасних методів діагностики раку молочної залози, сучасних інтелектуальних систем в сфері надання медичних послуг, проаналізовано переваги та недоліки існуючих систем. Було розроблено проєкт мобільного застосунку для пацієнтів, що надає можливість частково автоматизувати процес діагностики та продемонстровано практичну здійсненність та перспективність використання мобільних застосунків в сфері надання медичних послуг.

Практичне значення роботи полягає в потенціалі значного зменшення адміністративного тиску на медичний персонал та зменшенні кількості необхідних візитів до лікаря для проведення аналізів, які пацієнт може виконати власноруч або з використанням спеціалізованих пристроїв.

В подальшому слід очікувати, що подібний підхід до часткової автоматизації надання медичних послуг буде використовуватись у діагностиці більшої кількості захворювань.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Cancer - Screening and early detection. URL: https://www.who.int/europe/news-room/fact-sheets/item/cancer-screening-and-early-detection-of-cancer (дата звернення:10.01.2024).
2. Ruiz R.L., Duffy V.G. Automation in Healthcare Systematic Review // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer Science and Business Media Deutschland GmbH, 2021. Вип. 13097 LNCS. С. 111–124.
3. Овчар С.В., Чуйко Г.П. Інтелектуальна система моніторингу та діагностики раку молочної залози // Інформаційні технології і автоматизація – 2023 / Матеріали XVI міжнародної науково-практичної конференції. . Одеса: Видавництво ОНТУ. С. 429–430.
4. Nounou M.I. et al. Breast cancer: Conventional diagnosis and treatment modalities and recent patents and technologies supplementary issue: Targeted therapies in breast cancer treatment // Breast Cancer (Auckl). Libertas Academica Ltd., 2015. Вип. 9. С. 17–34.
5. Gøtzsche P.C., Jørgensen K.J. Screening for breast cancer with mammography // Cochrane Database of Systematic Reviews. John Wiley and Sons Ltd, 2013. Вип. 2013, № 6.
6. Torrisi L., Restuccia N., Torrisi A. Study of gold nanoparticles for mammography diagnostic and radiotherapy improvements // Reports of Practical Oncology and Radiotherapy. Urban and Partner, 2019. Вип. 24, № 5. С. 450–457.
7. Peters N.H.G.M. et al. Meta-analysis of MR imaging in the diagnosis of breast lesions // Radiology. Centre for Reviews and Dissemination (UK), 2008. Вип. 246, № 1. С. 116–124.
8. Mann R.M., Kuhl C.K., Moy L. Contrast-enhanced MRI for breast cancer screening // Journal of Magnetic Resonance Imaging. John Wiley and Sons Inc., 2019. Вип. 50, № 2. С. 377–390.

9. Jansen S.A. et al. DCEMRI of breast lesions: Is kinetic analysis equally effective for both mass and nonmass-like enhancement? // Med Phys. John Wiley and Sons Ltd, 2008. Вип. 35, № 7. С. 3102–3109.
10. Palmer M.L., Tsangaris T.N. Breast biopsy in women 30 years old or less // The American Journal of Surgery. Elsevier, 1993. Вип. 165, № 6. С. 708–712.
11. Zheng B., Yoon S.W., Lam S.S. Breast cancer diagnosis based on feature extraction using a hybrid of K-means and support vector machine algorithms // Expert Syst Appl. 2014. Вип. 41, № 4. С. 1476–1482.
12. Patrício M. et al. Using Resistin, glucose, age and BMI to predict the presence of breast cancer // BMC Cancer. BioMed Central Ltd., 2018. Вип. 18, № 1. С. 1–8.
13. Handbook of Hormones // Handbook of Hormones. Elsevier, 2021.
14. Jeyaraj A. Models of Information Technology Use: Meta-Review and Research Directions // Journal of Computer Information Systems. Taylor & Francis, 2023. Вип. 63, № 4. С. 809–824.
15. Koumpouros Y., Georgoulas A. A systematic review of mHealth funded R&D activities in EU: Trends, technologies and obstacles // Inform Health Soc Care. Taylor & Francis, 2020. Вип. 45, № 2. С. 168–187.
16. Imlawi J., Gregg D. Understanding the satisfaction and continuance intention of knowledge contribution by health professionals in online health communities // Inform Health Soc Care. Taylor and Francis Ltd, 2020. Вип. 45, № 2. С. 151–167.
17. Redelmeier D.A., Kraus N.C. Patterns in patient access and utilization of online medical records: Analysis of MyChart // J Med Internet Res. JMIR Publications Inc., 2018. Вип. 20, № 2. С. e8372.
18. Dinh-Le C. et al. Wearable Health Technology and Electronic Health Record Integration: Scoping Review and Future Directions // JMIR Mhealth Uhealth. JMIR Publications Inc., 2019. Вип. 7, № 9.

19. Udrea A. et al. Accuracy of a smartphone application for triage of skin lesions based on machine learning algorithms // Journal of the European Academy of Dermatology and Venereology. John Wiley & Sons, Ltd, 2020. Вип. 34, № 3. С. 648–655.
20. Jaworek-Korjakowska J., Kleczek C. ESkin: Study on the smartphone application for early detection of malignant melanoma // Wirel Commun Mob Comput. Hindawi Limited, 2018. Вип. 2018.
21. SkinVision response to: Algorithm based smartphone apps to assess risk of skin cancer in adults: systematic review of diagnostic accuracy studies | The BMJ. URL: https://www.bmj.com/content/368/bmj.m127/rr-2 (дата звернення:10.01.2024).
22. North F., Chaudhry R. Apple HealthKit and Health App: Patient Uptake and Barriers in Primary Care // https://home.liebertpub.com/tmj. Mary Ann Liebert, Inc. 140 Huguenot Street, 3rd Floor New Rochelle, NY 10801 USA , 2016. Вип. 22, № 7. С. 608–613.
23. iOS - Health - Apple. URL: https://www.apple.com/ios/health/ (дата звернення:10.01.2024).
24. Canedo E.D., Santos G.A. Factors affecting software development productivity: An empirical study // ACM International Conference Proceeding Series. Association for Computing Machinery, 2019. С. 307–316.
25. Nagle F. Open source software and firm productivity // Manage Sci. INFORMS Inst.for Operations Res.and the Management Sciences, 2019. Вип. 65, № 3. С. 1191–1215.
26. Amatya S., Kurti A. Cross-platform mobile development: Challenges and opportunities // Advances in Intelligent Systems and Computing. Springer Verlag, 2014. Вип. 231. С. 219–229.
27. Cross-platform mobile frameworks used by global developers 2022 | Statista. URL: https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/ (дата звернення:10.01.2024).

28. Flutter - Build apps for any screen. URL: https://flutter.dev/ (дата звернення:10.01.2024).
29. Dart overview | Dart. URL: https://dart.dev/overview (дата звернення:10.01.2024).
30. FAQ | Flutter. URL: https://docs.flutter.dev/resources/faq (дата звернення:10.01.2024).
31. flutter/examples/hello_world at 75228a59dacc24f617272f7759677e242bbf74ec · flutter/flutter. URL: https://github.com/flutter/flutter/tree/75228a59dacc24f617272f7759677e242bbf74ec/examples/hello_world (дата звернення:10.01.2024).
32. Analyze your build with the APK Analyzer | Android Studio | Android Developers. URL: https://developer.android.com/studio/debug/apk-analyzer (дата звернення:10.01.2024).
33. Reducing your app’s size | Apple Developer Documentation. URL: https://developer.apple.com/documentation/xcode/reducing-your-app-s-size (дата звернення:10.01.2024).
34. Filipova O., Vilão R. Backend Development // Software Development From A to Z. Apress, Berkeley, CA, 2018. С. 101–131.
35. Gropengießer F., Sattler K.-U. Database Backend as a Service: Automatic Generation, Deployment, and Management of Database Backends for Mobile Applications // Datenbank-Spektrum. Springer Science and Business Media LLC, 2014. Вип. 14, № 2. С. 85–95.
36. Firebase | Google’s Mobile and Web App Development Platform. URL: https://firebase.google.com/ (дата звернення:10.01.2024).
37. Supabase | The Open Source Firebase Alternative. URL: https://supabase.com/ (дата звернення:10.01.2024).
38. Firebase Documentation. URL: https://firebase.google.com/docs (дата звернення:10.01.2024).

39. Supabase Docs. URL: https://supabase.com/docs (дата звернення:10.01.2024).
40. Supabase is now HIPAA and SOC2 Type 2 compliant. URL: https://supabase.com/blog/supabase-soc2-hipaa (дата звернення:11.01.2024).
41. Summary of the HIPAA Security Rule | HHS.gov. URL: https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html (дата звернення:10.01.2024).
42. Smart Glucose Meters - Worldwide | Statista Market Forecast. URL: https://www.statista.com/outlook/hmo/digital-health/digital-treatment-care/connected-biometric-sensors/smart-glucose-meters/worldwide#users (дата звернення:10.01.2024).
43. Breitenbeck N., Brown A. Accuracy Assessment of a Blood Glucose Monitoring System for Self-Testing with Three Test Strip Lots Following ISO 15197:2013/EN ISO 15197:2015 // J Diabetes Sci Technol. SAGE Publications Inc., 2017. Вип. 11, № 4. С. 854–855.
44. Instant Blood Glucose Meter - Accu-Chek UK. URL: https://www.accu-chek.co.uk/blood-glucose-meters/instant (дата звернення:10.01.2024).
45. Glucose Profile | Bluetooth® Technology Website. URL: https://www.bluetooth.com/specifications/specs/glucose-profile-1-0-1/ (дата звернення:10.01.2024).
46. Intro to Bluetooth Generic Attribute Profile (GATT) | Bluetooth® Technology Website. URL: https://www.bluetooth.com/bluetooth-resources/intro-to-bluetooth-gap-gatt/ (дата звернення:10.01.2024).
47. Intro to Bluetooth Generic Access Profile (GAP) | Bluetooth® Technology Website. URL: https://www.bluetooth.com/bluetooth-resources/intro-to-bluetooth-generic-access-profile-gap/ (дата звернення:10.01.2024).
48. flutter_blue_plus URL: https://pub.dev/packages/flutter_blue_plus (дата звернення:10.01.2024).

ДОДАТОК А

ДОВІДКА

про перевірку на унікальність

пояснювальної записки кваліфікаційної магістерської роботи

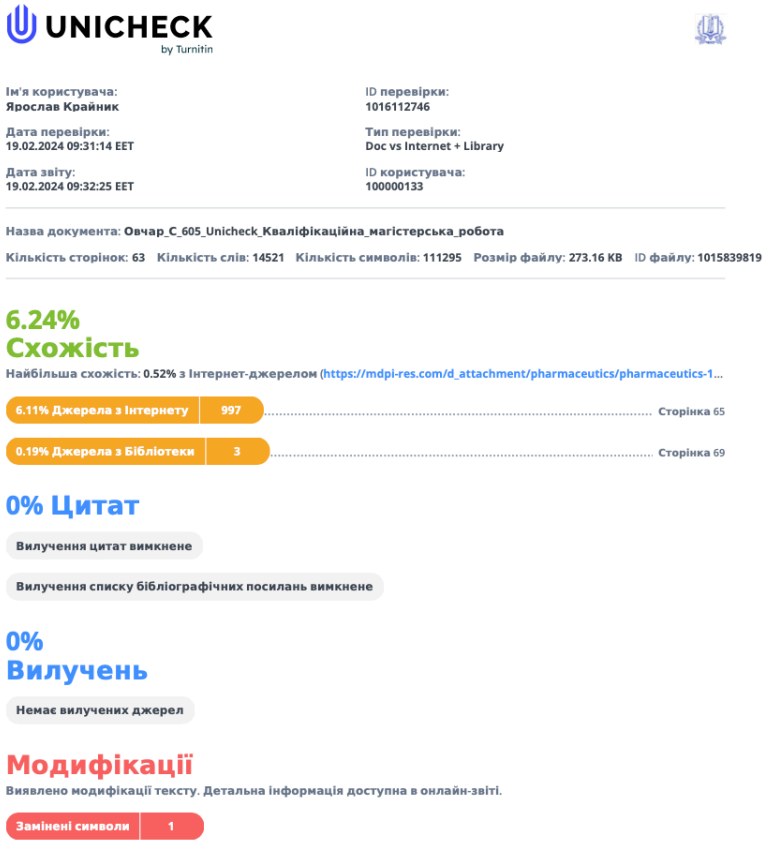
на тему: « Інтелектуальна система моніторингу та діагностики раку молочної залози»

студента спеціальності 123 «Комп'ютерна інженерія», групи 605м

Овчара Сергія Віталійовича

Перевірку тексту здійснено сервісом: онлайн-сервіс Unicheck

Результат перевірки тексту кваліфікаційної роботи: схожість складає 6,24 %.


UNICHECK
by Turnitin

Ім'я користувача: Ярослав Крайник
ID перевірки: 1016112746
Дата перевірки: 19.02.2024 09:31:14 EET
Тип перевірки: Doc vs Internet + Library
Дата звіту: 19.02.2024 09:32:25 EET
ID користувача: 100000133

Назва документа: Овчар_С_605_Unicheck_Кваліфікаційна_магістерська_робота

Кількість сторінок: 63 Кількість слів: 14521 Кількість символів: 111295 Розмір файлу: 273.16 KB ID файлу: 1015839819

6.24%
Схожість
Найбільша схожість: 0.52% з Інтернет-джерелом (https://mdpi-res.com/d_attachment/pharmaceutics/pharmaceutics-1...

6.11% Джерела з Інтернету 997 Сторінка 65

0.19% Джерела з Бібліотеки 3 Сторінка 69

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 1

Студент: Керівник:

канд. техн. наук, доцент

_____ С. В. Овчар
підпис ініціали, прізвище

_____ Я. М. Крайник
підпис ініціали, прізвище

Дата: «____» _____ 2024 р.

ДОДАТОК Б

Код програми

register_user/index.ts

```
import { SupabaseClient, User, createClient } from 'https://esm.sh/@supabase/supabase-
js@2.7.1'
import { TCreateUser, toTCreateUser } from "../_shared/user.ts";
import * as dbTables from "../_shared/db_tables.ts";
import { corsHeaders } from "../_shared/cors.ts";
console.log(`Function "register-user" up and running!`)
Deno.serve(async (req: Request) => {
  let supabaseClient: SupabaseClient | null = null;
  let toRegister: TCreateUser | null = null;
  let supabaseUser: User | null = null;
  try {
    supabaseClient = createClient(
      Deno.env.get('SUPABASE_URL') ?? '',
      Deno.env.get('SUPABASE_ANON_KEY') ?? '',
      // Create client with Auth context of the user that called the function.
      {
        global: {
          headers: { Authorization: req.headers.get('Authorization')! },
        },
      }
    )
  } catch (_error) {
    return new Response('Failed to initialize', {
      status: 500,
      headers: corsHeaders,
    });
  }
  try {
    const {
      data: { user },
    } = await supabaseClient!.auth.getUser()
    if (!user) {
      throw new Error('NoUserData');
    }
    supabaseUser = user!;
  } catch (error) {
    if (error.message === 'NoUserData') {
      return new Response('Not Authorized', {
        status: 401,
        headers: corsHeaders,
      });
    }
  }
  try {
    toRegister = toTCreateUser(await req.json());
  } catch (error) {
    return new Response(error.message, {
      status: 400,
      headers: { ...corsHeaders, 'Content-Type': 'application/json' },
    });
  }
  try {
    const { data, error } = await supabaseClient
      .from(dbTables.usersTableName)
      .insert({
        id: supabaseUser!.id,
        email: toRegister.email,
        name: toRegister.name,
        last_name: toRegister.last_name,
        date_of_birth: toRegister.date_of_birth,
```

```
        role: toRegister.role,
      })
      .select()
      .limit(1)
      .single();
    if (error) {
      return new Response(JSON.stringify({ error: error.message }), {
        headers: { ...corsHeaders, 'Content-Type': 'application/json' },
        status: 500,
      });
    }
    return new Response(JSON.stringify(data), {
      headers: { ...corsHeaders, 'Content-Type': 'application/json' },
      status: 200,
    })
  } catch (error) {
    return new Response(JSON.stringify({ error: error.message }), {
      headers: { ...corsHeaders, 'Content-Type': 'application/json' },
      status: 400,
    })
  }
})
```

get_user/index.ts

```
import { SupabaseClient, User, createClient } from 'https://esm.sh/@supabase/supabase-
js@2.7.1'
import { toTUser } from "../_shared/user.ts";
import * as dbTables from "../_shared/db_tables.ts";
import { corsHeaders } from "../_shared/cors.ts";
console.log(`Function "get-user" up and running!`)
Deno.serve(async (req: Request) => {
  if (req.method !== 'GET') {
    return new Response('Method not allowed', {
      status: 405,
      headers: corsHeaders,
    });
  }
  let supabaseClient: SupabaseClient | null = null;
  let supabaseUser: User | null = null;
  try {
    supabaseClient = createClient(
      Deno.env.get('SUPABASE_URL') ?? '',
      Deno.env.get('SUPABASE_ANON_KEY') ?? '',
      // Create client with Auth context of the user that called the function.
      {
        global: {
          headers: { Authorization: req.headers.get('Authorization')! },
        },
      }
    )
  } catch (_error) {
    return new Response('Failed to initialize', {
      status: 500,
      headers: corsHeaders,
    });
  }
  try {
    const {
```

```
      data: { user },
    } = await supabaseClient!.auth.getUser()
    if (!user) {
      throw new Error('NoUserData');
    }
    supabaseUser = user!;
  } catch (error) {
    if (error.message === 'NoUserData') {
      return new Response('Not Authorized', {
        status: 401,
        headers: corsHeaders,
      });
    }
  }
  try {
    const { data, error } = await supabaseClient
      .from(dbTables.usersTableName)
      .select('*').eq('id', supabaseUser!.id).limit(1).single();
    if (!data) {
      return new Response(JSON.stringify({ error: 'User not found' }), {
        headers: { ...corsHeaders, 'Content-Type': 'application/json' },
        status: 404,
      });
    }
    if (error) {
      return new Response(JSON.stringify({ error: error.message }), {
        headers: { ...corsHeaders, 'Content-Type': 'application/json' },
        status: 500,
      });
    }
    return new Response(JSON.stringify(toTUser(data)), {
      headers: { ...corsHeaders, 'Content-Type': 'application/json' },
      status: 200,
    });
  } catch (error) {
    return new Response(JSON.stringify({ error: error.message }), {
      headers: { ...corsHeaders, 'Content-Type': 'application/json' },
      status: 400,
    })
  }
})
```

get_doctor/index.ts

```
import { SupabaseClient, User, createClient } from 'https://esm.sh/@supabase/supabase-
js@2.7.1'
import { toTUser } from "../_shared/user.ts";
import * as dbTables from "../_shared/db_tables.ts";
import { corsHeaders } from "../_shared/cors.ts";
console.log(`Function "get-doctor" up and running!`)
Deno.serve(async (req: Request) => {
  if (req.method !== 'GET') {
    return new Response('Method not allowed', {
```

```
      status: 405,
      headers: corsHeaders,
    });
  }
  let supabaseClient: SupabaseClient | null = null;
  let supabaseUser: User | null = null;
  try {
    supabaseClient = createClient(
      Deno.env.get('SUPABASE_URL') ?? '',
      Deno.env.get('SUPABASE_ANON_KEY') ?? '',
      // Create client with Auth context of the user that called the function.
      {
        global: {
          headers: { Authorization: req.headers.get('Authorization')! },
        },
      }
    )
  } catch (_error) {
    return new Response('Failed to initialize', {
      status: 500,
      headers: corsHeaders,
    });
  }
  try {
    const {
      data: { user },
    } = await supabaseClient!.auth.getUser()
    if (!user) {
      throw new Error('NoUserData');
    }
    supabaseUser = user!;
  } catch (error) {
    if (error.message === 'NoUserData') {
      return new Response('Not Authorized', {
        status: 401,
        headers: corsHeaders,
      });
    }
  }
  try {
    const patientAssignedResponse = await supabaseClient
      .from(dbTables.patientsAssignedTableName)
      .select('*').eq('patient_id', supabaseUser!.id).limit(1).single();
    if (!patientAssignedResponse.data) {
      return new Response(JSON.stringify({ error: 'Patient is not assigned to a
doctor' }), {
        headers: { ...corsHeaders, 'Content-Type': 'application/json' },
        status: 404,
      });
    }
    if (patientAssignedResponse.error) {
```

```
      return new Response(JSON.stringify({ error:
patientAssignedResponse.error.message }), {
        headers: { ...corsHeaders, 'Content-Type': 'application/json' },
        status: 500,
      });
    }
    const patientAssigned = patientAssignedResponse.data;
    const doctorResponse = await supabaseClient
      .from(dbTables.usersTableName)
      .select('*').eq('id', patientAssigned.doctor_id).limit(1).single();
    if (!doctorResponse.data) {
      return new Response(JSON.stringify({ error: 'Doctor not found' }), {
        headers: { ...corsHeaders, 'Content-Type': 'application/json' },
        status: 404,
      });
    }
    if (doctorResponse.error) {
      return new Response(JSON.stringify({ error: patientAssigned }), {
        headers: { ...corsHeaders, 'Content-Type': 'application/json' },
        status: 500,
      });
    }
    const doctor = doctorResponse.data;
    return new Response(JSON.stringify(toTUser(doctor)), {
      headers: { ...corsHeaders, 'Content-Type': 'application/json' },
      status: 200,
    });
  } catch (error) {
    return new Response(JSON.stringify({ error: error.message }), {
      headers: { ...corsHeaders, 'Content-Type': 'application/json' },
      status: 500,
    })
  }
})
```

my_measurements/index.ts

```
import { SupabaseClient, User, createClient } from 'https://esm.sh/@supabase/supabase-js@2.7.1'


import { TUser, UserRole, toTUser } from "../_shared/user.ts";
import { toTPatientMeasurement, TCreatePatientMeasurement, toTCreatePatientMeasurement,
TFilterPatientMeasurement, toTFilterPatientMeasurement } from
"../_shared/patient_measurements.ts";
import * as dbTables from "../_shared/db_tables.ts";
import { corsHeaders } from "../_shared/cors.ts";

console.log(`Function "my-measurements" up and running!`)

Deno.serve(async (req: Request) => {
  if (req.method !== 'POST' && req.method !== 'PUT') {
    return new Response('Method not allowed', {
      status: 405,
      headers: corsHeaders,
    });
  }

  let supabaseClient: SupabaseClient | null = null;
  let supabaseUser: User | null = null;

  try {
```

```
    supabaseClient = createClient(
      Deno.env.get('SUPABASE_URL') ?? '',
      Deno.env.get('SUPABASE_ANON_KEY') ?? '',
      // Create client with Auth context of the user that called the function.
      {
        global: {
          headers: { Authorization: req.headers.get('Authorization')! },
        },
      }
    )
  } catch (_error) {
    return new Response('Failed to initialize', {
      status: 500,
      headers: corsHeaders,
    });
  }


  try {
    const {
      data: { user },
    } = await supabaseClient!.auth.getUser()

    if (!user) {
      return new Response('Not Authorized', {
        status: 401,
        headers: corsHeaders,
      });
    }

    supabaseUser = user!;
  } catch (error) {
    return new Response(JSON.stringify({ error: error.message }), {
      status: 500,
      headers: corsHeaders,
    });
  }

  try {
    const { data, error } = await supabaseClient
      .from(dbTables.usersTableName)
      .select('*').eq('id', supabaseUser!.id).limit(1).single();

    if (error) {
      return new Response(JSON.stringify({ error: error.message }), {
        headers: { ...corsHeaders, 'Content-Type': 'application/json' },
        status: 500,
      });
    }

    if (!data || data!.role !== UserRole.patient) {
      return new Response('Forbiden', {
        headers: { ...corsHeaders, 'Content-Type': 'application/json' },
        status: 403,
      });
    }

    const user = toTUser(data);


    if (req.method === 'PUT') {
      return await newMeasurement(req, supabaseClient, user);
    } else {
      return await getMeasurements(req, supabaseClient, user);
    }

  } catch (error) {
    return new Response(JSON.stringify({ error: error.message }), {
      headers: { ...corsHeaders, 'Content-Type': 'application/json' },
      status: 400,
    })
  }
})

async function newMeasurement(req: Request, supabaseClient: SupabaseClient, user: TUser):
Promise<Response> {
```

```
  let measurement: TCreatePatientMeasurement;

  try {
    measurement = toTCreatePatientMeasurement(await req.json());
  } catch (error) {
    return new Response(error.message, {
      status: 400,
      headers: { ...corsHeaders, 'Content-Type': 'application/json' },
    });
  }

  try {
    const insertResponse = await supabaseClient
      .from(dbTables.patientMeasurementsTableName)
      .insert({
        patient_id: user.id,
        type: measurement.type,
        value: measurement.value,
        unit: measurement.unit,
        created_by: user.id,
      })
      .select('*')
      .limit(1)
      .single();

    if (insertResponse.error) {
      return new Response(JSON.stringify({ error: insertResponse.error.message }), {
        headers: { ...corsHeaders, 'Content-Type': 'application/json' },
        status: 500,
      });
    }

    return new Response(JSON.stringify(toTPatientMeasurement(insertResponse.data)), {
      headers: { ...corsHeaders, 'Content-Type': 'application/json' },
      status: 201,
    });

  } catch (error) {
    return new Response(JSON.stringify({ error: error.message }), {
      headers: { ...corsHeaders, 'Content-Type': 'application/json' },
      status: 500,
    });
  }

}

async function getMeasurements(req: Request, supabaseClient: SupabaseClient, user: TUser):
Promise<Response> {
  let filter: TFilterPatientMeasurement | null = null;

  try {
    filter = toTFilterPatientMeasurement(await req.json());
  } catch (_) {
    return new Response('Invalid Filter', {
      status: 400,
      headers: { ...corsHeaders, 'Content-Type': 'application/json' },
    });
  }

  try {
    let query = supabaseClient
      .from(dbTables.patientMeasurementsTableName)
      .select('*')
      .eq('patient_id', user.id)
      .order('created_at', { ascending: false });

    if (filter && (filter.type || filter.limit)) {
      if (filter.type) {
        query = query.eq('type', filter.type);
      }
      if (filter.limit) {
        query = query.limit(filter.limit);
      }
    }

    const { data, error } = await query;
```

```
    if (error) {
      return new Response(JSON.stringify({ error: error.message }), {
        headers: { ...corsHeaders, 'Content-Type': 'application/json' },
        status: 500,
      });
    }

    // deno-lint-ignore no-explicit-any
    const mappedData = data.map((item: any) => toTPatientMeasurement(item));

    return new Response(JSON.stringify(mappedData), {
      headers: { ...corsHeaders, 'Content-Type': 'application/json' },
      status: 200,
    });

  } catch (error) {
    return new Response(JSON.stringify({ error: error.message }), {
      headers: { ...corsHeaders, 'Content-Type': 'application/json' },
      status: 500,
    });
  }
}
```

auth_bloc.dart

```
import 'dart:async';
import 'package:bcres/common/common.dart';
import 'package:bcres/features/auth/auth.dart';
import 'package:equatable/equatable.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:supabase/supabase.dart';
part 'auth_event.dart';
part 'auth_state.dart';
class AuthBloc extends Bloc<AuthEvent, AuthState> {
  final AuthRepository _authRepository;
  StreamSubscription? _authSubscription;
  AuthBloc({
    required AuthRepository authRepository,
  })  : _authRepository = authRepository,
        super(AuthInitial()) {
    on<AuthSignInRequested>(_onSignInRequested);
    on<AuthSignUpRequested>(_onSignUpRequested);
    on<AuthSignOutRequested>(_onSignOutRequested);
    on<AuthUserDataChanged>(_onUserDataChanged);
    _authSubscription = _authRepository.userStream.listen(_onUserChanged);
  }
  factory AuthBloc.fromContext(BuildContext context) => AuthBloc(
        authRepository: context.read<AuthRepository>(),
      );
  Future<void> _onSignInRequested(AuthSignInRequested event, Emitter<AuthState> emit)
async {
    emit(const SigningIn());
   if (event.email.isEmpty || event.password.isEmpty) {
      emit(const SignInFailed('Email and password cannot be null'));
      return;
    }
    try {
      final response = await _authRepository.loginWithEmail(
        email: event.email,
        password: event.password,
      );
      if (response == AuthLoginStatus.success) {
        event.onSuccess?.call();
        return;
      }
      // if we somehow get here, then we failed to authenticate
      emit(SignInFailed(defaultSignInError));
      event.onError?.call();
    } catch (e) {
      emit(
```

```
        SignInFailed(
          decodeSignInError(e),
        ),
      );
      event.onError?.call();
    }
  }
  Future<void> _onSignUpRequested(AuthSignUpRequested event, Emitter<AuthState> emit)
async {
    emit(const SigningUp());
    if (event.email.isEmpty) {
      emit(const SignUpFailed(LocaleKeys.auth_error_email));
      return;
    }
    if (event.password.isEmpty) {
      emit(const SignUpFailed(LocaleKeys.auth_error_password));
      return;
    }
    try {
      final response = await _authRepository.signUpWithEmail(
        email: event.email,
        password: event.password,
      );
      if (response == AuthLoginStatus.success) {
        event.onSuccess?.call();
        // }
        emit(const SignedUp());
        return;
      }
      // if we somehow get here, then we failed to authenticate
      emit(SignUpFailed(defaultSignUpError));
      event.onError?.call();
    } catch (e) {
      emit(
        SignUpFailed(
          decodeSignUpError(e),
        ),
      );
      event.onError?.call();
    }
  }
  Future<void> _onSignOutRequested(AuthSignOutRequested event, Emitter<AuthState>
emit) async {
    emit(const SigningOut());
    try {
      final response = await _authRepository.logout();
      if (response == AuthLoginStatus.loggedOut) {
        emit(const SignedOut());
        event.onSuccess?.call();
      }
    } catch (e) {
      emit(const SignOutFailed());
      event.onError?.call();
    }
  }
  void _onUserChanged(User? event) => add(AuthUserDataChanged(userData: event));
  Future<void> _onUserDataChanged(AuthUserDataChanged event, Emitter<AuthState> emit)
async {
    if (event.userData == null) {
      emit(const SignedOut());
    } else {
      emit(
        Authenticated(event.userData!),
      );
    }
  }
}
```

patient_measurement_cubit.dart

```
import 'dart:async';
import 'package:bcres/common/common.dart';
import 'package:bcres/features/patient/patient.dart';
import 'package:bcres/features/user_details/user_details.dart';
import 'package:equatable/equatable.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
part 'patient_measurements_state.dart';
class PatientMeasurementsCubit extends Cubit<PatientMeasurementsState> {
  final PatientApiService _patientApiService;
  StreamSubscription? _userDetailsSubscription;
  PatientMeasurementsCubit({
    required UserDetailsBloc userDetailsBloc,
    required PatientApiService patientApiService,
  })  : _patientApiService = patientApiService,
        super(const PatientMeasurementsEmpty()) {
    _userDetailsSubscription =
userDetailsBloc.stream.listen(_handleUserBlocUpdate);
    if (userDetailsBloc.state is UserDetailsWithDataState) {
      _changeUser((userDetailsBloc.state as
UserDetailsWithDataState).userDetails.id);
    }
  }
  factory PatientMeasurementsCubit.fromContext(BuildContext context) {
    return PatientMeasurementsCubit(
      userDetailsBloc: context.read<UserDetailsBloc>(),
      patientApiService: context.read<PatientApiService>(),
    );
  }
  void _changeUser(String? userId) {
    if (userId == null) {
      emit(const PatientMeasurementsEmpty());
      return;
    }
    load(userId);
  }
  void _handleUserBlocUpdate(UserDetailsState userDetailsState) {
    final userDetails = userDetailsState is UserDetailsWithDataState ?
userDetailsState.userDetails : null;
    if (userDetails == null && !state.isEmpty) {
      _changeUser(null);
      return;
    }
    if (userDetails != null &&
        userDetails.role.isPatient &&
        (!state.isEmpty ||
            !(state is PatientMeasurementsWithUserIdState &&
                (state as PatientMeasurementsWithUserIdState).userId ==
userDetails.id))) {
      _changeUser(userDetails.id);
    }
  }
  Future<void> load(String userId) async {
    emit(PatientMeasurementsLoading(userId: userId));
    try {
      final measurements = await
_patientApiService.getPatientMeasurementsSummary();
      emit(
        PatientMeasurementsLoaded(
          userId: userId,
          measurements: measurements,
```

```
        ),
      );
    } on Exception catch (e) {
      emit(PatientMeasurementsLoadFailed(e.toString()));
    }
  }
  Future<void> addNewMeasurement(CreatePatientMeasurement toCreate) async {
    if (state is! PatientMeasurementsLoaded) {
      return;
    }
    final userId = (state as PatientMeasurementsLoaded).userId;
    final oldMeasurements = (state as
PatientMeasurementsLoaded).measurements;
    final newValue = PatientMeasurement(
      id: 'inprogress',
      patientId: userId,
      type: toCreate.type,
      value: toCreate.value,
      unit: toCreate.unit,
      createdBy: userId,
      createdAt: DateTime.now(),
    );
    // optimisticUpdate
    final newSummary = oldMeasurements.copyWith(
      height: newValue.type.isHeight ? newValue : null,
      weight: newValue.type.isWeight ? newValue : null,
      glucose: newValue.type.isGlucose ? newValue : null,
      resistin: newValue.type.isResistin ? newValue : null,
    );
    emit(PatientMeasurementsUploading(
      userId: userId,
      measurements: newSummary,
    ));
    try {
      final created = await
_patientApiService.createPatientMeasurement(toCreate);
      final newSummary = oldMeasurements.copyWith(
        height: created.type.isHeight ? created : null,
        weight: created.type.isWeight ? created : null,
        glucose: created.type.isGlucose ? created : null,
        resistin: created.type.isResistin ? created : null,
      );
      emit(PatientMeasurementsLoaded(
        userId: userId,
        measurements: newSummary,
      ));
    } catch (e) {
      emit(PatientMeasurementsUploadFailed(
        userId: userId,
        measurements: oldMeasurements,
      ));
    }
  }
}
```

ДОДАТОК В

Публікації за темою роботи

Міністерство освіти і науки України
Одеський національний технологічний університет
Інститут комп`ютерної інженерії, автоматизації, робототехніки та програмування ім.П.Н.Платонова

«ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ І АВТОМАТИЗАЦІЯ – 2023»

МАТЕРІАЛИ
XVI МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ

19 - 20 ЖОВТНЯ 2023 р.

м.ОДЕСА

УДК 004.4

ІНТЕЛЕКТУАЛЬНА СИСТЕМА МОНІТОРИНГУ ТА ДІАГНОСТИКИ РАКУ МОЛОЧНОЇ ЗАЛОЗИ

Овчар С. В., Чуйко Г. П. (ovcharka132@gmail.com, genchuiko@gmail.com)
Чорноморський національний університет імені Петра Могили (Україна)

В тезах розглядається мобільного застосунку для інтелектуального моніторингу та діагностики раку молочної залози, який автоматизує збір даних пацієнта, дозволяє лікарям вести онлайн-консультації та слідкувати за динамікою захворювання, його сприяння ранньому виявленню захворювання та вплив мобільних застосунків на ефективність лікування і адміністративне навантаження на лікарів.

У сучасному медичному контексті діагностика та моніторинг раку молочної залози є важливим аспектом забезпечення довголіття та якості життя жінок. Зростаюча потреба в оперативному виявленні та контролі змін дозволяє розглядати новітні технологічні рішення як ключ до вдосконалення медичних підходів. Впровадження мобільних застосунків може значущо підвищити швидкість та ефективність виявлення патологій, надаючи можливість проводити регулярний моніторинг у домашніх умовах та віддалено передавати дані спеціалістам [1]. Використання сучасних цифрових технологій в медицині відкриває нові горизонти для підвищення ефективності діагностики, лікування та профілактики.

У цій роботі розглядається можливість розробки мобільного застосунку, що призначений для підтримки процесів діагностики та моніторингу раку молочної залози. Застосунок повинен інтегруватись з сучасними датчиками, що можуть відстежувати різноманітні показники стану здоров'я пацієнта, забезпечуючи точний та оперативний збір даних.

Мобільні застосунки дозволяють забезпечити індивідуалізований підхід до кожного пацієнта, адаптуючи рекомендації та плани лікування на основі персональних даних, вони також можуть слугувати засобом освіти пацієнтів, надаючи їм доступ до інформації про їх стан здоров'я, рекомендацій щодо лікування та профілактики. Автоматизований збір даних з датчиків може виявляти патологічні зміни на ранніх стадіях, значно підвищуючи шанси на успішне лікування. Автоматизоване ведення медичних записів, сповіщення та інші функції можуть значно зменшити адміністративне навантаження на лікарів [2].

Рисунок 1 – Функціонал застосунку в залежності від ролі користувача

Мобільний застосунок, спрямований на діагностику раку молочної залози, пропонує ряд ключових функцій (див. рис. 1):

– Інтерфейс для лікарів: Лікарі можуть переглядати медичні дані пацієнтів, отримувати звіти про стан здоров'я, стежити за динамікою хвороби та надсилати рекомендації пацієнтам;

– Інтерфейс для пацієнтів: Пацієнти мають можливість вести щоденник свого здоров'я, отримувати відгуки від лікарів та консультуватися з ними в режимі онлайн;

– Автоматизований збір даних з медичних пристроїв (датчиків): Застосунок може синхронізуватися з зовнішніми пристроями, які використовує пацієнт (ваги, глюкометр тощо), для автоматичного збору даних про стан здоров'я, що можуть вказувати на розвиток або прогресування захворювання;

– Збереження даних медичних оглядів та досліджень: Лікарі мають можливість завантажувати результати медичних досліджень в базу даних;

– Інтелектуальна діагностика: За допомогою вбудованих алгоритмів застосунок аналізує збережені дані про стан здоров’я пацієнта виявляє патологічні зміни та інформує про потенційні ризики.

Для розробки мобільного застосунку з діагностики раку молочної залози заплановано використовувати технології Flutter, Express.js та Firebase.

Flutter — це open-source фреймворк від Google, призначений для розробки мобільних застосунків. Його мультиплатформність дозволяє створювати додатки одразу для iOS і Android з одного кодового базису, що прискорює процес розробки та знижує вартість. Flutter також відомий своєю високою продуктивністю та легкістю створення адаптивних дизайнів.

Express.js — це мінімалістичний та гнучкий фреймворк для веб-застосунків на Node.js. Його простота та гнучкість сприяє швидкому розгортанню веб-сервера, а підтримка великої кількості додаткових бібліотек значно полегшує розробку архітектури.

Firebase — платформа розробки від Google, яка пропонує різноманітні інструменти та сервіси. Серед основних особливостей Firebase можна відзначити інструменти для аутентифікації користувачів та бази даних, які дозволяють здійснювати синхронізацію даних у реальному часі між користувачами та сервером. Також Firebase легко масштабується з ростом кількості користувачів та відмінно інтегрується з іншими сервісами та платформами, такими як мобільні застосунки.

У контексті розробки медичних застосунків, комбіноване використання вищенаведених технологій надає можливість швидко та ефективно створювати надійні, масштабовані та безпечні рішення, які можуть адаптуватися до потреб користувачів та бізнесу в медичному секторі, підвищуючи ефективність діагностики, лікування та профілактики.

Висновки. Сучасний мобільний застосунок для моніторингу та діагностики раку молочної залози, розроблений на основі передових технологій, відіграє ключову роль у ранньому виявленні захворювань. Автоматизований збір даних, онлайн-консультації з лікарями та моніторинг динаміки покращують якість діагностики. Такі інновації можуть стати вирішальними в системі охорони здоров'я майбутнього.

Список використаної літератури

[1] Cruz, F. O. A. M., Vilela, R. A., Ferreira, E. B., Melo, N. S., & Dos Reis, P. E. D. (2019). Evidence on the use of mobile apps during the treatment of breast cancer: systematic review. *JMIR mHealth and uHealth*, *7*(8), e13245.

[2] Mamoun, R., Nasor, M. and Abulikailik, S.H., 2021, February. Design and development of mobile healthcare application prototype using Flutter. In 2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE) (pp. 1-6). IEEE.

[3] Flutter - Build apps for any screen [Online]. Available: https://flutter.dev [Accessed: 01.10.2023].

[4] Express - Node.js web application framework [Online]. Available: https://expressjs.com [Accessed: 01.10.2023]..

[5] Firebase | Google’s Mobile and Web App Development Platform [Online]. Available: https://firebase.google.com [Accessed: 01.10.2023].