Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська дипломна робота на тему:

ПРОГРАМНИЙ МОДУЛЬ МОНІТОРИНГУ ПРОЦЕСІВ ЖИТТЄДІЯЛЬНОСТІ ТВАРИННИЦЬКОГО СКЛАДУ ФЕРМЕРСЬКОГО ГОСПОДАРСТВА

Виконав: студент 4 курсу, групи 2КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Ярощук А. О.
(прізвище та ініціали)
Керівник БДР: д.т.н., проф. каф. КН
Іванчук Я. В.
(прізвище та ініціали)
«07» 06 2024 р.

Рецензент: д.т.н., проф., зав. каф. КСУ
Ковтун В. В.
(прізвище та ініціали)
«07» 06 2024 р.

Допущено до захисту
Зав. кафедри проф., д.т.н. Яровий А.А.
«07» 06 2024 р.

Вінниця ВНТУ - 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
«07» 05 2024р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Ярощук Анастасії Олегівни

1. Тема роботи: Програмний модуль моніторингу процесів життєдіяльності тваринницького складу фермерського господарства.
Керівник роботи: Іванчук Ярослав Володимирович, д.т.н, проф. кафедри КН
затверджені наказом вищого навчального закладу "11" 05 2024 року №__
2. Строк подання студентом роботи 27.05.2024р.
3. Вихідні дані до роботи: персональні дані про співробітників; мова програмування SQL, Dart та Flutter; максимальна кількість введеня тварин – 1500 од.; максимальна кількість введення показників здоров'я – 8 од.; кількість сторінок інтерфейсу користувача – не менше 5 од.;
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): аналіз предметної області, використання ветеринарних методів та засобів для ранньої діагностики захворювань сільськогосподарських тварин, огляд існуючих програмних засобів для відстеження показників здоров'я тварин, реалізація алгоритмів створення програмного модуля для моніторингу процесів життєдіяльності тварин фермерського господарства; проєктування програмних модулю моніторингу процесів життєдіяльності тварин фермерського господарства; програмна реалізація та тестування програмного модуля моніторингу процесів життєдіяльності тварин фермерського господарства.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): типова схема запиту та відповіді від сервера; типова схема архітектури програмного модулю відстеження; загальний алгоритм роботи та структурна схема програмного модулю моніторингу показників здоров'я тварин фермерського господарства; UML-діаграма бази даних програмного модуля відстеження показників тварин; схема алгоритму моніторингу та візуалізації показників здоров'я тварин; принципова схема роботи архітектурного шаблону MVC

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завда прий
1-3	Іванчук Я. В., д.т.н., проф.. каф. КН		

7. Дата видачі завдання 07.05.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Прим
		початок	закінчення	
1	Визначення об'єкта, предмета, мети та задачі подальшого дослідження	06.05.24	10.05.24	
2	Аналіз предметної області, існуючих методів, алгоритмівта інструментів для створення програмного модуля для моніторингу процесів життєдіяльності тварин фермерського господарства ;	11.05.24	16.05.24	
3	Проєктування програмних програмного модуля для моніторингу процесів життєдіяльності тварин фермерського господарства;	17.05.24	24.05.24	
4	Програмна реалізація програмного модуля для моніторингу процесів життєдіяльності тварин фермерського господарства	25.05.24	03.06.24	
5	Оформлення пояснювальної записки	04.06.24	06.0624	

Студентка _____ **Ярощук А. О.**
(підпис) *(прізвище та ініціали)*

Керівник роботи _____ **Іванчук Я. В.**
(підпис) *(прізвище та ініціали)*

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 63 сторінки формату А4, на яких є 16 рисунки, список використаних джерел містить 24 найменувань.

Дана бакалаврська робота присвячена розробці та впровадженню програмного модуля для моніторингу процесів життєдіяльності тварин фермерського господарства. Основною метою роботи є є підвищення ефективності в управлінні комплексом та забезпечення можливості швидко вжити заходів при виявленні відхилень у показниках під час відстеження її здоров’я моніторинг показників здоров'я тварин на фермерському господарстві з допомогою програмного модуля. Була розроблена структурна схема для опису функціонування модулю, схема алгоритму роботи функцій виявлення відхилень показників від норми та візуалізації графіків, UML-діаграма алгоритму моніторингу процесів життєдіяльності тварин фермерського господарства. Flutter та Dart використовуються для розробки мобільного застосунку та забезпечення взаємодії з користувачем. SQLite використовується для зберігання та керування базою персональних даних працівників, обліку тварин та їх показників здоров’я. Архітектура даного модуля реалізована таким чином, що дозволяє забезпечити розділення логіки, представлення та обробки даних.

Результатом роботи є функціональний і надійний програмний модуль, який дозволяє ефективно використовувати економічні ресурси та зменшує час реагування фахівця на відхилення показників, тим самим дозволяє швидко запобігти розвитку тієї чи іншої хвороби тварини.

Ключові слова: інтелектуальний модуль, агросфера, моніторинг стану тварин, автоматизація процесів, фермерське господарство, інтеграція.

ABSTRACT

The Bachelor's thesis consists of 63 pages in A4 format, including 16 figures, and the reference list contains 24 sources.

This Bachelor's thesis is devoted to the development and implementation of a software module for monitoring the life processes of farm animals. The main goal of the work is to increase the efficiency of managing the complex and provide the ability to quickly take action when deviations in indicators are detected during health monitoring of animals on the farm using the software module. A structural diagram was developed to describe the functioning of the module, an algorithm scheme for detecting deviations from the norm and visualizing graphs, and a UML diagram of the algorithm for monitoring the life processes of farm animals. Flutter and Dart are used to develop the mobile application and interact with the user. SQLite is used for storing and managing a database of employee personal data, animal records, and their health indicators. The architecture of this module is implemented in such a way as to provide separation of logic, presentation, and data processing.

The result of the work is a functional and reliable software module that allows efficient use of economic resources and reduces the response time of specialists to deviations in indicators, thereby quickly preventing the development of various diseases in animals.

Keywords: intelligent module, agrosphere, animal health monitoring, process automation, farm, integration

ЗМІСТ

ВСТУП

Актуальність досліджень. Україна має значний аграрний потенціал, тому тема автоматизації процесів на фермерському господарстві є дуже актуальною. Зі зростанням кількості даних на фермерських господарствах та проблем ефективності методів управління, існує нагальна потреба в технологічних рішеннях, які спрощують збір й аналіз даних та покращують процеси прийняття рішень [2].

Однією з головних проблем, які вирішить програмний модуль є вчасне виявлення ознак хвороб та незвичайних станів у тварин. Шляхом моніторингу цих параметрів фермери зможуть реагувати швидко та ефективно, запобігаючи подальшому поширенню захворювання та мінімізуючи втрати. Крім того, модуль допоможе оптимізувати використання ресурсів, таких як корм, вода та ліки, за рахунок аналізу годівлі та репродуктивності тварин. Це дозволить фермерам ефективніше управляти виробництвом та зменшити витрати [3].

Програмний модуль допоможе покращити добробут тварин та підвищити їхню продуктивність, оскільки він дозволить виявляти та вирішувати проблеми вчасно, забезпечуючи комфортні умови існування.

Реалізація програмного модуля моніторингу процесів життєдіяльності тваринницького складу фермерського господарства дозволяє господарям фермерського господарства покращити управління господарством, збільшити продуктивність та знизити витрати, що в цілому призведе до покращення їхнього бізнесу.

Метою дослідження є підвищення ефективності визначення показників здоров’я свійських тварин за допомого розробки програмного модуля моніторингу процесів життєдіяльності тваринницького складу фермерського господарства, що дозволить оперативно вживати заходи по зниженню рівня захворюванності.

Об’єктом дослідження є процес розробки програмного модуля моніторингу процесів життєдіяльності тваринницького складу фермерського господарства.

Предметом дослідження є алгоритми та програмні засоби моніторингу показників здоров’я тваринницького складу фермерського господарства, його функціональні можливості та інтеграція із біосенсорами, що забезпечують збір даних.

Задачі дослідження:

1. Провести аналіз предметної області дослідження, методів та засобів збору данних процесів життєдіяльності тваринницького складу фермерського господарства.
2. Розробка і проєктування структури програмного модуля автоматизованого відстеження показників здоров’я тварин, отриманих із спеціальних біосенсорів.
3. Розробка бази даних програмного модуля моніторингу показників здоров’я тварин на фермерському господарстві.
4. Розробка алгоритму функціонування програмного модуля моніторингу показників здоров’я тварин на фермерському господарстві.
5. Програмна реалізація модуля моніторинг показників здоров’я тварин на фермерському господарстві.
6. Розробка інструкції користувача програмного модуля моніторинг показників здоров’я тварин на фермерському господарстві.

Апробація роботи.

Робота апробувалась на конференції:

- LIII Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2024)

Публікації: електронні тези на тему «Програмний модуль моніторингу процесів життєдіяльності тваринницького складу фермерського господарства» [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз класифікації тваринницьких ферм та особливості технологій утримання тварин

Тваринництво – невід’ємна складова сільського господарства, відіграє ключову роль у забезпеченні якісною продукцією та розвитку економіки країни. Поєднуючи рослинництво, воно оптимізує використання земельних ресурсів та забезпечує необхідні компоненти для годівлі та догляду за тваринами. Розвиток технологій дозволяє підвищувати ефективність виробництва та покращує якість продукції [2]. Необхідно збалансувати розвиток тваринництва і рослинництва для максимального використання ресурсів. Успішне тваринницьке підприємство вимагає ефективного управління, а також розвитку каналів збуту для продукції. Працюючи над цими аспектами, можна забезпечити стабільний та ефективний розвиток тваринництва у сільському господарстві.

Залежно від виду тварин, що утримуються, розрізняють ферми та комплекси великої рогатої худоби, свинарські, вівчарські, птахівницькі, звірівницькі. Розглянемо кожен вид ферми окремо.

У вівчарстві на фермах та комплексах утримують і вирощують овець для одержання вовни, каракулевих смужок, овчини, м’яса та молока [10]. Вівчарство вимагає уважного управління та врахування різноманітних аспектів для забезпечення їхнього здоров'я, добробуту та продуктивності. Ось деякі ключові особливості утримання овець: належно побудований комлекс для тварин має важливе значення для їхнього комфорту та захисту від негативних погодних умов. Це може включати в себе стійла, приміщення для відгодівлі, загони для випасу та інші спеціалізовані структури. Якість годівлі - це ключовий фактор для здоров'я та їх продуктивності.

Особливості догляду за їх здоров'ям - це регулярні ветеринарні огляди, вакцинації та профілактичні заходи проти хвороб та паразитів є важливими для забезпечення здоров'я овець. Особливу увагу слід приділяти захисту від

гельмінтів. Е свою чергу, ефективне управління репродукцією та поголів'ям допомагає забезпечити стійкий ріст та розвиток стада. Це включає в себе вибір кращих порід для розведення, контроль сезонності розпліду та відбір статей для покриття [2].

Утримання свиней з точки зору здоров'я є ключовим аспектом фермерської діяльності. Забезпечення оптимальних умов життя, правильної годівлі та регулярного ветеринарного нагляду є необхідним для підтримки здоров'я та добробуту тварин. Свині повинні мати доступ до чистої води, належного житла та відповідної годівлі, що містить у собі необхідні поживні речовини та вітаміни. Регулярні ветеринарні перевірки та вакцинації допомагають запобігти захворюванням та контролювати розповсюдження інфекційних захворювань серед стада. Особлива увага приділяється захисту від паразитів та виконанню вчасним профілактичним заходам [3]. Ефективне управління поголів'ям, контроль репродуктивного здоров’я та відбір статей для розведення також важливі для збереження здоров'я та підтримки продуктивності стада. Усі ці заходи спрямовані на забезпечення оптимальних умов для свиней, щоб вони росли здоровими, сильними та продуктивними, що в свою чергу сприяє успішності фермерського господарства.

У молочному скотарстві створюються такі ферми та комплекси: змішані із закінченим виробничим циклом спеціалізовані, на яких, крім корів, утримують телят тільки в період випоювання молоком; спеціалізовані на вирощуванні телиць для комплектування молочного стада [3, 10]. Утримання молочного скоту з точки зору здоров’я є критично важливим аспектом для досягнення високої продуктивності та довговічності тварин.

Забезпечення оптимальних умов життя, правильної годівлі та регулярного моніторингу стану здоров'я є ключовими факторами. Кращі умови життя включають в себе належно побудований комплекс з відповідною вентиляцією та освітленням, доступ до свіжої та чистої води, а також достатній простір для руху та відпочинку. Годівля скоту повинно бути збалансованим та містити всі необхідні живильні речовини, вітаміни та мінерали. Регулярні ветеринарні

перевірки, вакцинації та профілактичні заходи допомагають запобігти захворюванням та контролювати розповсюдження інфекцій серед тварин [2, 4]. Крім того, моніторинг фізичного стану, включаючи вагу, температуру тіла та активність, дозволяє вчасно виявляти проблеми зі здоров'ям та приймати необхідні заходи.

Ефективне еправління поголів'ям, включаючи вибір племінних тварин та контроль розведення, також сприяє збереженню здоров'я стада та підтримує його продуктивність на високому рівні. Всі заходи спрямовані на забезпечення оптимального здаров'я та добробуту молочного скоту, щоб вони могли продуктивно забезпечувати молоко та приносити користь фермерському господарству. Утримання ферми кіз з точки зору здоров'я включає в себе ряд важливих аспектів, які допомагають забезпечити оптимальні умови для тварин і підтримувати їхнє добробут. Належне житло та утримання, забезпечення належної годівлі, регулярний ветеринарний нагляд і вакцинації, контроль за паразитами та ефективне управління репродукцією - це ключові аспекти догляду за козами.

Забезпечення чистої стайні, доступу до свіжої води та достатнього простору для руху допомагає уникнути захворювань та стресу. Годівля кіз має бути збалансованим і містити необхідну кількість білків, вітамінів та мінералів для підтримки здоров'я та продуктивності. Регулярні медичні огляди і вакцинації допомагають попередити захворювання і контролювати їх поширення серед стада. Контроль за паразитами, такими як гельмінти, також є важливим аспектом утримання кіз. Крім того, ефективне управління репродукцією, включаючи контроль за покриттям і вибір племінних кіз, допомагає забезпечити стійкий ріст стада та підтримує його продуктивність. Всі ці заходи спрямовані на забезпечення оптимального здоров'я та добробуту кіз, щоб вони могли продуктивно забезпечувати молоко та інші корисні продукти для фермерського господарства [2].

Отже, в ході виконання аналізу різних видів ферм та особливостей утримання кожної з тварин встановимо, що найефективнішим програмний

модуль для відстеження показників здоров'я тварин буде у молочному скотарстві. Ця програма буде розроблена з урахуванням особливостей утримання корів, їхніх фізіологічних потреб та характеристик молочного виробництва. Вона повинна забезпечувати можливість моніторингу ключових показників, таких як годівля, фізичний стан, репродуктивне здоров'я та медичний стан, з метою вчасного виявлення потенційних проблем та прийняття необхідних заходів.

Додатково, програма повинна бути гнучкою, щоб враховувати різні методи утримання корів та специфічні потреби різних ферм. Крім того, важливим аспектом є забезпечення можливості аналізу даних та надання рекомендацій для оптимального управління стадом корів, що сприятиме підвищенню продуктивності, здоров'я та добробуту корів у молочному скотарстві.

1.2 Аналіз використання ветеринарних методів та засобів для ранньої діагностики захворювань сільськогосподарських тварин

На управлінні сільськогосподарськими підприємствами значно впливають технологічні досягнення, які зменшують фізичну працю, витрати та відходи, одночасно збільшуючи врожайність і прибутки. Моніторинг вегетативних реакцій у режимі реального часу (наприклад, частоти дихання, варіабельності серцевого ритму та частоти серцевих скорочень, артеріального тиску, змін у периферичному кровотоці) і захисних рефлексів за допомогою інноваційного біосенсорного обладнання може допомогти зрозуміти, як утримання, годівля та генотип впливають на тварин стійкість до стресових факторів. Ці датчики можуть сприяти пізнанню факторів, що впливають на добробут тварин, і створенню засобів захисту (наприклад, техніки вирощування, відбору генотипу), які підвищують добробут худоби та тварин-компаньйонів. Датчики можуть відстежувати харчову поведінку, поведінку при жуванні, pH рубця, температуру рубця, температуру тіла, поведінку несучості, активність тварин, а також місцезнаходження або розміщення тварин [4]. Існує багато гаджетів, які можуть

вимірювати температуру тіла, поведінку та рухи тварин. Датчики та переносні технології можна вставляти в тварин, щоб виявляти компоненти їхніх тілесних рідин, наприклад піт [7]. На корів також надягають комерційно доступні нашийники з біосенсорами для моніторингу естрального циклу [3, 9].

Використання технологій для оцінки фізіологічних, поведінкових і продуктивних маркерів на окремих тваринах з метою ідентифікації цікавих подій включено в процес прецизійного моніторингу молочних продуктів [4]. Аналіти молока можна використовувати як біомаркери для виявлення захворювань або репродуктивного статусу. Наприклад, DeLaval Herd Navigator визначає кількість прогестерону в молоці. Програма також рекомендує ідеальний час для осіменіння, називає тварин, вагітність яких потребує підтвердження, позначає ранні аборти та перераховує корів із ризиком виникнення кіст і тривалого анеструсу. Інші доїльні роботи, такі як Lely Astronaut можуть визначати аналіти молока та електропровідність молока. Біомаркери крові є корисними індикаторами здоров'я тварин, хоча вони мають обмежене економічне використання [11]. Вони можуть надати багато інформації, особливо тому, що біомаркери можуть виявляти субклінічні стадії захворювань, навіть якщо корова виглядає повністю здоровою і не має видимих ознак хвороби.

Альтернативою біомаркерам крові можуть бути біомаркери молока [8]. В даний час на фермах широко використовуються сенсорні прилади для визначення вмісту жиру і білка в молоці. Відповідно до системи доїння кожної ферми використовується унікальна система датчиків. Ці датчики надають інформацію про здоров'я та плодючість великої рогатої худоби. За допомогою аналізаторів молока можна відслідковувати естральний цикл і репродективні показники у дійних корів [3]. Існують різні аналізатори, які можуть працювати на різних принципах, одні використовують хімічні індикатори, а інші базуються на спектроскопії.

Пов'язані з маститом втрати молока у молочних тварин є економічно значущими. Моніторинг концентрації соматичних клітин (КСК) у молоці є найпоширенішим методом виявлення маститу, особливо при його субклінічних

формах. Коли значення КСК перевищують ліміт, значення молока різко падає. Як результат, експерти вважають, що рівень КСК є важливим параметром для оцінки здоров'я вимені. Незважаючи на відсутність клінічних проявів, субклінічний мастит характеризується збільшенням КСК у молоці. Це є корисним індикатором здоров'я вимені, оскільки він підраховує кількість соматичних клітин (головним чином десквамованих епітеліальних клітин, макрофагів і нейтрофілів) у молоці [5]. Окрім використання як критерію для відбору молочних корів, які менш сприйнятливі до маститу, наявність КСК у коров'ячому молоці є добре встановленим показником запалення молочних залоз, яке тісно пов'язане з наявністю інфекції молочної залози. Виявилося, що КСК є корисним індикатором зменшення кількості молока внаслідок субклінічного маститу, який присутній, коли кількість клітин перевищує 200 000 на мілілітр [3]. За допомогою методів фарбування клітин і мікроскопії концентрацію КСК можна оцінити на лабораторному рівні. Однак і підходи займають багато часу та потребують спеціального обладнання та персоналу. Сучасні методи визначення КСК при виявленні молока є набагато менш трудомісткими.

Автоматичні пристрої для виявлення маститу є прикладами нових діагностичних процесів, які адаптуються до польових умов, є простими та можуть швидко надавати результати. Існує багато типів обладнання, наприклад перевірки молока, є такі пристрої: лічильник Fossomatic (Данія), детектор маститу Dramiski/ Wykrywacz (Польща), лічильник клітин DeLaval (Швеція), детектор маститу Afimilk (Ізраїль), тест UdderCheck (США) і тест PortaSCC® (США) [4]. Вони покладаються або на виявлення фізико-хімічно-біологічних змін у молоці чи вимені, або на оцінку біомаркерів у рідинах організму (молоко, сироватка), пов'язаних із маститом [8].

Було продемонстровано, що діагностична здатність інфрачервоної термографії (IRT) порівнянна з каліфорнійським тестом на мастит, і вона також відрізняє випадки клінічного маститу від випадків субклінічного маститу. Як наслідок, IRT має потенціал перетворитися на діагностичний інструмент, який є

одночасно зручним і портативним [8].

Контроль фертильності потребує тісної координації між фермерами та ветеринарами, систематичного вивчення сільськогосподарської документації та надійних клінічних даних. Крім того, низьку фертильність можна вважати ознакою поганого здоров’я та благополуччя. Розведення є невід'ємною складовою тваринництва. Виявлення фази овуляції у великої рогатої худоби має вирішальне значення для визначення найкращого періоду для штучного осіменіння. Наявність прогестерону передбачає наявність функціонуючого жовтого тіла. У результаті його десятиліттями використовували як біомаркер репродуктивної ефективності. Прогестерон у молоці є потенційним індикатором репродуктивного статусу дійних корів, оскільки прогестерон транспортується з крові в молоко [3]. Таким чином, датчики прогестерону можуть бути корисними сенсорними системами.

Дослідники вже давно зацікавлені у виявленні захворювань шляхом ідентифікації летких органічних сполук (ЛОС). Тварини та люди як видихають, так і виділяють ЛОС зі своїми виділеннями з диханням, кров’ю, калом, шкірою, сечею та вагінальними виділеннями [9]. Такі гази, як водень (H2) і метан (CH4), а також леткі органічні молекули, такі як жирні кислоти, які можуть служити біомаркерами для метаболічних і патологічних процесів. ЛОС, такі як кетонові тіла, етанол, метанол та екзогенні сполуки, зазвичай пов’язані з рівнем глюкози в крові. Аналіз ЛОС використовувався для вивчення респіраторної хвороби великої рогатої худоби, бруцельозу, туберкульозу великої рогатої худоби, хвороби Джона, кетоацидозу та нормальної фізіології рубця великої рогатої худоби [9].

Більшість біосенсорів для аналізу метаболітів поту були створені з метою моніторингу здоров’я людини. Вони використовувалися для вимірювання рівня лактату та концентрації солі, а також були зроблені портативними (у формі пояса) для вимірювання поту. Крім того, цей датчик може бути модифікований для використання при вимірюванні потовиділення тварин, зокрема як індикатор фізичного стресу у тварин.

Збір слини для визначення захворювань та інших біохімічних маркерів фізіологічного здоров’я є привабливою не інвазивною альтернативою забору крові [9]. Оскільки слина містить як локальні, так і системні компоненти, вона є корисним джерелом інформації про системні процеси, що відбуваються в організмі, що дозволяє оцінити фізіологічний або патологічний стан організму. Однак використання слини в діагностиці хвороб тварин потребує детального дослідження її білкового складу в різних ситуаціях [8].

Техніка особливо корисна для спостереження за тваринами та діагностики захворювань, оскільки вважається, що взяття крові у тварин є стресовим фактором і може вплинути на біохімічні параметри, що вимірюються. Біомаркери слини можуть бути корисними для кількох цілей, включаючи раннє виявлення та діагностику захворювань, підтримку прийняття рішень щодо догляду за тваринами та моніторинг прогресування захворювання . Існують інші діагностичні застосування для слини, такі як дослідження біомаркерів слини для виявлення раку порожнини рота. Слина може бути цінним діагностичним зразком, що містить можливі ознаки фізіологічних і патологічних станів, таких як стан вагітності великої рогатої худоби. Аналіти слини були досліджені на предмет їх зв’язку з кульгавістю в іншому дослідженні.

Також передбачалося, що корови можуть змінювати певні частинки в слині, деякі з них можуть відображати поліпшення кульгавості після терапії.

Портативні електронні пристрої моніторингу мають потенціал трансформувати інтенсивне великомасштабне молочне виробництво шляхом моніторингу та керування коровами на індивідуальній основі. З початку 2000-х років спостерігалося помітне збільшення кількості опублікованих досліджень, присвячених застосуванню електронних пристроїв моніторингу для використання в комерційному фермерському середовищі [3].

Для розпізнавання рухів щелепи, які характеризують поведінку великої рогатої худоби на випасі, можна використовувати три різні типи біосенсорів. Це датчики електроміографії, механічні датчики/датчики тиску та акустичні датчики [9]. Тому поведінка великої рогатої худоби на випасі потребує

ретельного спостереження за кожною коровою на основі трьох ключових факторів: положення корови, аналіз її пози та рухів корови, зокрема її ходи та руху щелеп. Кількість часу, який тварина проводить з опущеною вниз головою, додається до записаних даних датчика для розрахунку часу пасіння . Наприклад, безперервне спостереження за рухами щелеп може виявити інформацію про денні звички випасу, розлади здоров'я тварин і дефіцит корму [9].

Тварини коригують свою поведінку у відповідь на стреси, соціальні зміни та зміни навколишнього середовища, що може спричинити захворювання. Через працю, необхідну для постійного моніторингу великих груп тварин, відстеження такої поведінки у великому масштабі стає неможливим. У поєднанні з належною інтерпретацією вихідних даних технології переносних датчиків дозволяють одночасно вимірювати фізіологічні параметри стада в реальному часі у великому масштабі.

Як наслідок, технології датчиків пропонують перевагу перед традиційними системами на основі стада, оскільки дані з датчиків можуть бути оцінені миттєво, що забезпечує короткий час реакції. Активність може бути знижена у великої рогатої худоби, ураженої кульгавістю або такими захворюваннями, як респіраторні захворювання великої рогатої худоби (BRD).

Збереження енергії для метаболічних витрат імунної системи та непрямих ефектів лихоманки та запальних реакцій на інфекцію було запропоновано як біологічну основу для зниження активності у хворих тварин [10]. Існує багато технологій для аналізу руху, руху та поведінки, таких як акселерометр, крокоміри та система глобального позиціонування (GPS).

Акселерометри використовувалися в системах молочного скотарства для виявлення таких захворювань, як мастит, а також для виявлення тічки та проблем з пересуванням. Зміни показань акселерометра також можна використовувати для створення контрольного рівня активності, який згодом можна записати як розраховану кількість кроків або інші індекси руху, наприклад коефіцієнти активності. Акселерометри набули популярності в дослідженнях м'ясного скотарства, оскільки вони дозволяють безперервно та довгостроково вивчати

рухливість і поведінку тварини.

Використовуючи відстеження в режимі реального часу за системою глобального позиціонування (GPS) і біологічні технології, можна здійснювати дистанційний моніторинг тварин, щоб шукати будь-які ознаки хвороби або занепокоєння щодо їхнього благополуччя.

Різноманітні дослідження, проведені протягом останнього десятиліття, довели корисність телеметричних пристроїв GPS для аналізу поведінки великої рогатої худоби в поєднанні з іншими пристроями/датчиками. Поєднання GPS-нашийників із датчиками активності дає ефективний метод відстеження місцезнаходження тварин, що пасуться, і одночасного визначення поведінки тварин. Для точного визначення положення об’єкта в певній місцевості були створені системи визначення місця розташування в реальному часі. Незважаючи на те, що було проведено мало досліджень щодо поведінки та пересування кульгавих корів на пасовищі, технологія GPS може бути корисною для покращення автоматичної діагностики кульгавості систем на основі пасовища. За даними Riaboff та ін., сильно кульгаві корови витрачають у 4,5 рази менше часу на випас і майже вдвічі більше часу відпочивають у лежачому положенні, ніж їхні здорові корови [6]. Крім того, GPS використовується для прогнозування поведінки жуйних і визначення місця їх розташування на пасовищах. Ця техніка геолокації з високою частотою та низьким рівнем помилок, незважаючи на неадекватність GPS для надійного прогнозування поведінки. Використання даних акселерометрів і пристроїв GPS разом є підходом, який може виявитися корисним у процесі дослідження того, як корови взаємодіють з навколишнім середовищем. Ці складні сценарії можуть включати тепловий стрес, фізичний стрес, виснаження ресурсів, обмежений доступ до пасовищ та інші подібні ситуації. Ці характеристики можуть бути використані для підвищення ефективності існуючих датчиків виявлення кульгавості в системах на пасовищах.

Приклад застосування біосенсорів для виявлення аномалій на основі певної симптоматики відображено в таблиці 1.1.

Таблиця 1.1 Приклад визначення стану або відхилень тварини за допомогою біосенсорів

Статус тварини/відхилення	Спосіб визначення	Аналіз
Тільність	Роботи для доїння, радіоімунноаналіз, ферментовий імунноаналіз, акселерометри, педометри.	Кількість гормону прогестерону в молоці, активність, температура.
Кульгавість	Триаксіальні акселерометри, педометри, відеоспостереження, акселерометри, датчик румінації.	Рухливість, поведінка при годівлі, активність та час лежання були пов'язані з кульгавістю.
Мастит	Обробка зображень, спектроскопія, електрична провідність, біосенсори, датчики SCC (кількості соматичних клітин), тривимірні акселерометри, педометри, спектроскопія	Температура, поведінка при лежанні, харчування, якість молока (жирність, білок, електрична провідність), час румінації, кількість соматичних клітин (SCC), pH молока, виробництво молока
Ацедоз	Тривимірні акселерометри, датчики кутової швидкості, pH-метр, доїльні роботи.	Якість молока (жир, білок), активність, поведінка румінації, ходьба, поведінка під час годівлі.
Метрит/Ендометрит	Триосний акселерометр, електронна система годування.	Час їжі, час пиття, румінація, активність, час лежання
Кетоз	Камери з тривимірним зображенням, спектроскопія, роботизовані доїльні апарати, акселерометри.	Оцінка стану тіла, BHB (бета-гідроксибутират), якість молока (жир, білок), активність, поведінка румінації

Отже, згідно аналізу використання ветеринарних методів та інструментів для ранньої діагностики захворювань сільськогосподарських тварин, можна зазначити, що ці технології відіграють ключову роль у забезпеченні здоров'я та добробуту тварин на фермах. Вони дозволяють вчасно виявляти захворювання, навіть на субклінічній стадії, та приймати ефективні заходи для їх лікування та запобігання поширенню. Технології моніторингу, такі як біосенсори та датчики, дозволяють отримувати дані про фізіологічні параметри тварин у реальному часі, що дозволяє оперативно реагувати на будь-які зміни у їх стані здоров'я. Крім того, інноваційні методи аналізу молока та інших біоматеріалів надають

можливість оцінювати рівень маститу та інших захворювань з високою точністю.

Застосування таких методів дозволяє підвищити продуктивність господарства, зменшити втрати від захворювань та підвищити якість продукції. Однак важливо пам'ятати, що успішне використання цих технологій потребує не лише доступу до сучасного обладнання, але й навчання фахівців у сфері ветеринарної медицини та технологій. Також важливо постійно вдосконалювати та адаптувати методи для врахування специфіки кожного господарства та виду тварин.

Загалом, використання ветеринарних методів та інструментів для ранньої діагностики захворювань сільськогосподарських тварин сприятиме підвищенню ефективності та стабільності сільськогосподарського виробництва.

1.3 Сучасні програмні засоби для відстеження показників здоров’я тварин молочного скотарства

Сучасні програмні засоби для відстеження показників здоров'я тварин молочного скотарства дозволяють ефективно контролювати та аналізувати різноманітні параметри для забезпечення їхнього оптимального стану. Ці програмні засоби зазвичай включають функціонал для ведення обліку молочної продуктивності, графіків лактації, моніторингу здоров'я, контролю репродуктивних процесів, та інші. Вони надають можливість вести детальний облік кожної тварини, її історію хвороб та лікування, а також автоматизувати процеси внесення та аналізу даних. Завдяки цим засобам, фермери можуть оперативно реагувати на будь-які відхилення в стані здоров'я тварин та приймати необхідні заходи для підтримки їхнього здоров'я та продуктивності.

Спочатку розглянемо програму DelPro Companion від всесвітньовідомої шведської компанії DeLaval отримує сповіщення та надає в режимі реального часу дані про виробництво молока та стан здоров’я тварин з вашої системи VMS (система добровільного доїння) або AMR (автоматична роторна доїльна

установка) на пристрій [11]. Розглянемо інтерфейс даного додатку, який вказано на рисунку 1.1.

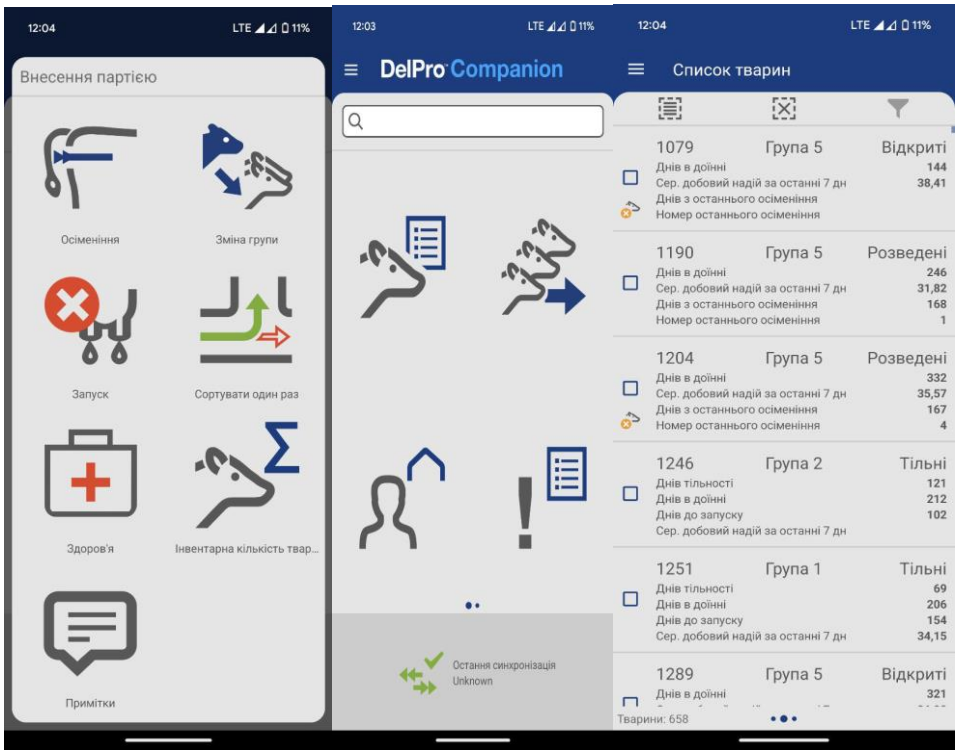

Рисунок 1.1 – Загальний вигляд інтерфейсного вікна програми DelPro

Подана система містить велику кількість фільтрів за багатьма параметрами, що дозволяє спеціалісту проаналізувати й ухвалити найоптимальніше рішення в кожному випадку, вчасно його провести. Наприклад, відібрати корів, яких вигідніше осіменити на другий цикл або визначити кандидаток на вибракування зі стада, а з них відібрати корів із певним рівнем надоїв. Після ухвалення рішення спеціалістом, DelPro генерує план дій на день і фахівцям потрібно тільки провести роботу згідно з планом - пролікувати, вибракувати, осіменити тощо.

На перший погляд, DelPro виглядає як потужний інструмент для оптимізації управління молочним господарством, проте є кілька недоліків:

1) Залежність від якості даних: Правильність рішень, прийнятих системою DelPro, залежить від якості та достовірності введених в неї даних. Якщо дані введені неправильно або неповністю, це може призвести до ухвалення неточних рішень або навіть помилок.

2) Залежність від фахівців: При всій потужності системи DelPro, успішне керування господарством залежить від кваліфікації та досвіду фахівців, які працюють з системою. Навіть з передовими технологіями, людський фактор є важливим. Також багато даних потрібно вносити вручну.

3) Перевантаження інформацією: Велика кількість зібраної системою інформації може бути перевантаженою для деяких користувачів, особливо для тих, хто не звик до роботи з обширними обліковими системами.

Хоча DelPro має багато переваг, розглянуті аспекти важливі для збалансованого управління та використання цієї системи.

На українському ринку також, часто використовується, таке програмне забезпечення, як UNIFORM від міжнародної компанії Uniform Agri, головний офіс якої знаходиться у Нідерландах [12]. Варто відзначити, що беручи до уваги досвід співпраці у Європі з організаціями ідентифікації та лабораторіями, що проводять аналіз молока, зараз в Україні почала налагоджуватись подібна співпраця, що в найближчому майбутньому надасть можливість обміну даними між лабораторією, агентством ідентифікації та реєстрації тварин і безпосередньо фермою. А саме, такі показники аналізу молока як: жирність, вміст білку, соматичних клітин, сечовини будуть надсилатись автоматично безпосередньо в програму на господарстві. Ця опція дасть можливість точно аналізувати ситуацію на господарстві щодо годівлі та маститу.

Подана система має такі переваги як простота у використанні, багатофункціональні інтерфейси, надійність бази даних, що розрахована на 15 тисяч корів, контроль показників відтворення через спеціальні звіти та аналізи, моніторинг стану здоров'я тварин (кількість соматичних клітин, використання медикаментів, реєстрування захворювань та профілактичних заходів). Завдяки даній програмі фахівці можуть сформувати власні звіти по будь-яким необхідним показникам (продуктивність, здоров'я, відтворення та інші), а також є можливість створювати нагадування, які дії потрібно виконувати та з якими групами тварин, що є дуже зручним. Інтерфейс програми розглянемо на рисунку 1.2.

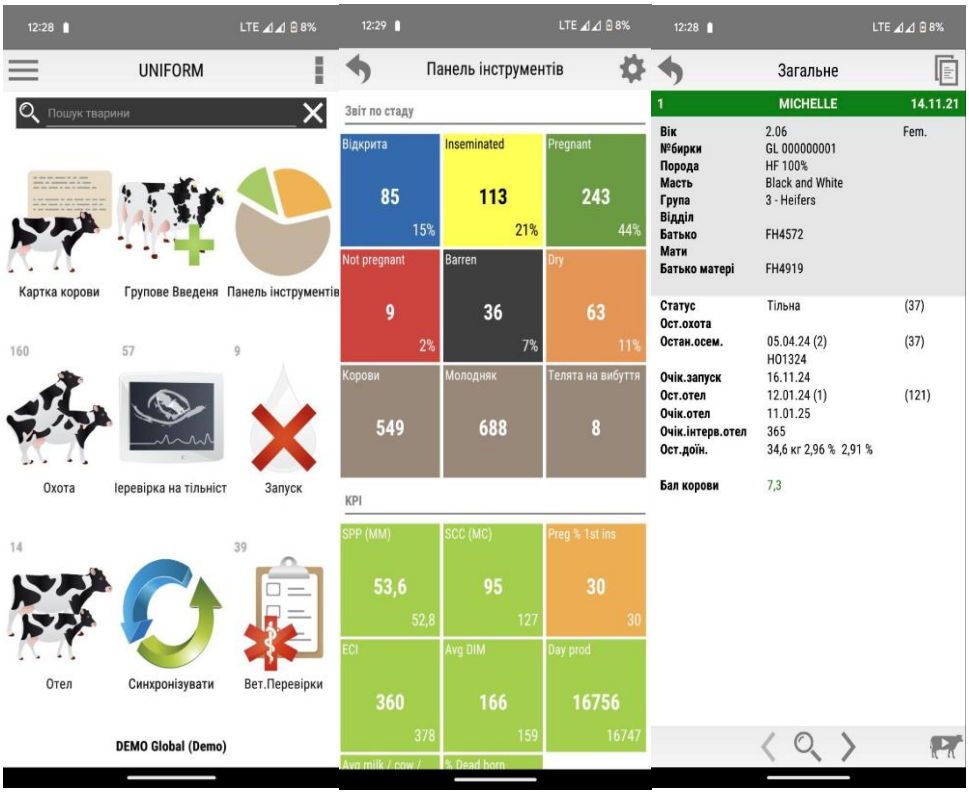

Рисунок 1.2 – Загальний вигляд інтерфейсного вікна програми UNIFORM

Програма UNIFORM має безліч переваг, але є деякі недоліки, які можна виокремити:

1) Велика кількість даних вноситься фахівцями вручну для ведення лікування та встановлення діагнозу тварин молочного скотарства.

2) Залежність від специфічного обладнання: Сумісність з певними доїльними залами може створювати обмеження для ферм, які використовують інше обладнання або планують змінити його у майбутньому.

Кожен з цих додатків має свій унікальний інтерфейс та переваги, але найбільшим недоліком кожної з програм: велика кількість даних вводиться вручну. Особливо для ведення лікування та встановлення діагнозу тварин молочного скотарства фахівцям необхідно вручну вводити велику кількість даних, що може бути трудомістким і збільшувати ризик помилок.

1.4 Висновок до розділу 1

Згідно проведеного аналізу предметної області, можна зробити висновок, що введення великої кількості даних вручну може бути проблемою для багатьох програм, включаючи DelPro та UNIFORM. Це може бути часовим і ресурсом затратним процесом, особливо на великих фермах з великою кількістю тварин. Ручне введення даних також збільшує ризик помилок та неточностей, оскільки е може призвести до затримок у веденні даних, а також збільшити ризик помилок та неточностей. Для подолання цих проблем можна розглянути використання інтелектуальних модулів для автоматизації процесу моніторингу та аналізу даних. Наприклад, реалізація інтелектуального модуля для моніторингу показників здоров'я тварин молочного скотарства може суттєво полегшити та автоматизувати процес ведення даних. Цей модуль може використовувати дані, зібрані з різних датчиків, що встановлені на тваринах або в їхньому середовищі. Датчики в свою чергу, можуть вимірювати температуру тіла, пульс, рухи тварин та інші параметри, які можуть вказувати на виникнення певних проблем зі здоров'ям.

За допомогою інтелектуального аналізу цих даних модуль може робити ранню діагностику хвороб та інших проблем на фермі. Він може автоматично виявляти відхилення в показниках та надсилати сповіщення фахівцям, що дозволить їм швидше реагувати та приймати відповідні заходи для запобігання поширенню захворювань та збереження здоров'я тварин.

2 РОЗРОБКА ТА ПРОЄКТУВАННЯ ПРОГРАМОГО МОДУЛЯ ДЛЯ МОНІТОРИНГУ ПОКАЗНИКІВ ЗДОРОВ'Я ТВАРИН

2.1 Розробка архітектури програмного модулю

Розробка архітектури програмного модуля для моніторингу процесів життєдіяльності тваринницького складу фермерського господарства полягає у створенні програмного забезпечення, яке дозволить відстежувати та аналізувати різноманітні аспекти функціонування тваринницького господарства.

Для побудови ефективного програмного модулю найдоцільніше використовувати клієнт-серверну архітектуру, включає кілька ключових компонентів і етапів.

Клієнт-серверна архітектура [13] передбачає розподіл завдань між клієнтськими пристроями (мобільними додатками) та сервером, який обробляє та зберігає дані. На стороні клієнта, мобільний додаток має забезпечувати інтуїтивно зрозумілий і функціональний інтерфейс для фермерів і ветеринарів. Клієнтська частина програми відповідає за збір даних від датчиків, відображення поточних станів тварин, графіків та діаграм. Користувачі можуть отримувати сповіщення про відхилення в стані здоров'я корів, переглядати історію даних, налаштовувати параметри моніторингу та зв'язуватися з ветеринарами через додаток. Зобразимо структуру даної архітектури на рисунку 2.1.

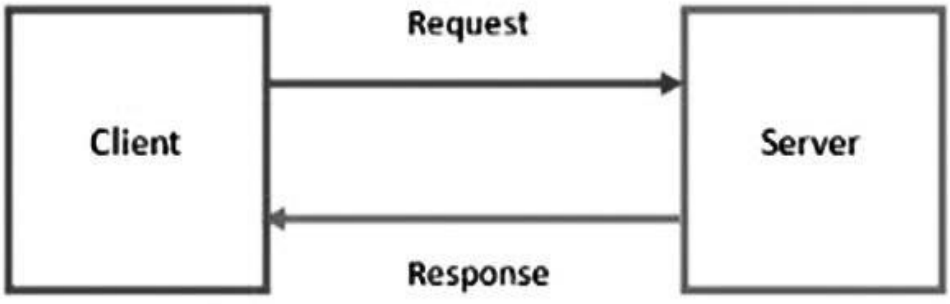

Рисунок 2.1 – Типова схема запиту та відповіді від сервера

Розробка програмного модуля для моніторингу процесів життєдіяльності тваринницького складу фермерського господарства на основі клієнт-серверної архітектури в мобільному додатку включає кілька ключових компонентів і етапів. Клієнт-серверна архітектура передбачає розподіл завдань між клієнтськими пристроями (мобільними додатками) та сервером, який обробляє та зберігає дані. На стороні клієнта, мобільний додаток має забезпечувати інтуїтивно зрозумілий і функціональний інтерфейс для фермерів і ветеринарів. Клієнтська частина програми відповідає за збір даних від датчиків, відображення поточних станів тварин, графіків та діаграм. Користувачі можуть отримувати сповіщення про відхилення в стані здоров’я корів, переглядати історію даних, налаштовувати параметри моніторингу та зв’язуватися з ветеринарами через додаток.

Подана система включатиме в себе засоби для збору та обробки даних про здоров'я, годівлю, розвиток та інші параметри тварин, що дозволить власникам фермерських господарств ефективно керувати та оптимізувати свою діяльність. Рисунок 2.2 надає фундаментальне уявлення про те, як система працюватиме.

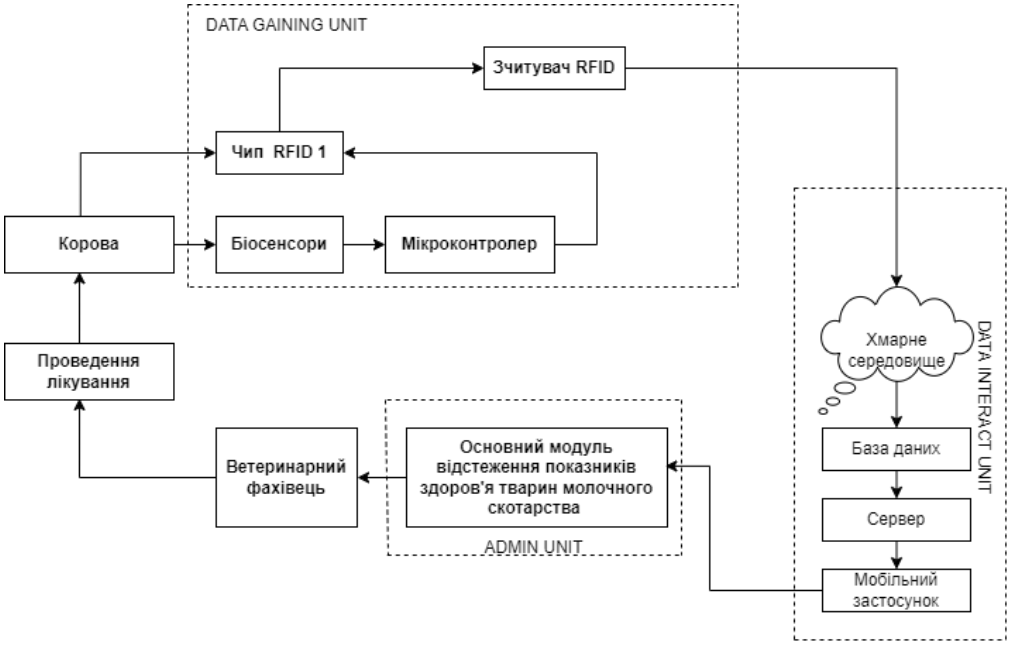

Рисунок 2.2 – Типова схема архітектури програмного модулю відстеження показників здоров’я тварин молочного скотарства

Згідно рисунку 2.2 моніторинг стану здоров'я здійснюється за допомогою датчика температури, датчика вологості та монітора серцевого ритму, які є у спеціалізованому нашийнику та відбуватиметься інтеграція та синхронізація даних із доїльним обладнанням, яке є на фермерському господарстві. Наприклад, датчик температури відповідає за перевірку температури корів, щоб будь-які відхилення або аномалії могли бути швидко виявлені. Монітор пульсу відстежує частоту серцевих скорочень тварини, щоб у разі підвищення пульсу можна було вжити заходів. Ця інформація також може бути використана для прогнозування стану тварини, що допомагає фермерам швидко реагувати на зміни в поведінці. Моніторинг частоти пульсу на регулярній основі дозволяє оцінити стан здоров'я тварини та вжити необхідних заходів для забезпечення її здоров'я.

Дані збираються всіма датчиками періодично, залежно від потреб користувачів. Ці дані передаються до центрального процесора, який обробляє інформацію для надання важливих відомостей, що потім аналізуються фахівцями. Дані зберігаються на мобільному додатку, доступ до якого мають уповноважені особи. Ці дані захищені, оскільки повністю зберігаються в резервних копіях у мобільному додатку. Дані публікуються у хмарі, що дозволяє фермерам та іншим користувачам використовувати цю інформацію для підвищення продуктивності і та ефективності, використовуючи вже наявну інформацію для досягнення хороших результатів.

Блок збору даних (Data gaining unit) складається з різних біомедичних датчиків, таких як датчик температури тіла, пульсовий датчик, датчик вологості, датчик частоти серцевих скорочень, які підключаються до мікроконтролера. Ці блоки збору даних отримують інформацію і роблять її доступною для будь-якого блоку управління даними, а також для блоку взаємодії з даними.

Датчики використовуються для основних і загальних автоматичних вимірювань багатьох медичних параметрів. Такі типи датчиків здоров'я встановлюються на тіло худоби і постійно фіксують стан тіла тварин. Ці сигнали порівнюються зі стандартними межами нормальних значень, встановленими як початкові точки в блоці управління даними. Якщо блок управління даними

виявляє значні або незвичайні зміни у певних тварин, він може звернутися до найближчого ветеринара. Якщо ветеринар не може бути присутнім у найближчій клініці, то за допомогою блоку управління на базі інтернеті розумних речей відповідальна особа може передати графіки спеціалісту. Спостерігаючи за цими графіками, лікар може надати інформацію про стан здоров'я тварини та відповідне лікування, яке слід застосувати до худоби без присутності лікаря під час надзвичайної ситуації. Приклад загального алгоритму роботи поданої системи моніторингу показників здоров’я тварин побудований на основі вище запропонованої архітектури мобільного додатку вказано на рисунку 2.3.

Рисунок 2.3 – Загальний алгоритм роботи програмного модулю моніторингу показників здоров’я тварин фермерського господарства

Розробка архітектури програмного модуля для моніторингу стану корів у фермерському господарстві передбачає створення комплексної системи, що забезпечує збір, зберігання, аналіз та візуалізацію даних про тварин. Система включає блок збору даних, який використовує різноманітні біометричні датчики для вимірювання температури, пульсу, активності та інших параметрів. Дані з датчиків передаються до центрального процесора, який проводить їх первинну обробку та виявляє аномалії. Оброблені дані зберігаються у базі даних, яка організована для швидкого доступу та ефективного оновлення.

Модуль аналізу даних використовує алгоритми для інтерпретації та прогнозування стану здоров'я корів. Інтерфейс користувача забезпечує зручний доступ до даних для фермерів та ветеринарів через мобільний додаток. Модуль сповіщення інформує користувачів про будь-які відхилення від норми. Інтеграція з хмарою забезпечує збереження та доступність даних з будь-якого місця, а модуль безпеки гарантує захист даних від несанкціонованого доступу.

Центральний процесор також відповідає за агрегування даних з різних джерел і виконання комплексних розрахунків для виявлення прихованих аномалій і прогнозування можливих проблем зі здоров'ям. Крім того, система включає модулі для управління налаштуваннями і правами доступу, що дозволяє адміністраторам ферми персоналізувати систему під конкретні потреби та забезпечувати різні рівні доступу для користувачів.

Система підтримує функцію історію аналізу даних, що дозволяє відстежувати зміни у здоров’ї корів протягом тривалого часу та приймати більш обґрунтовані рішення. Впровадження цієї архітектури сприятиме оптимізації операційних процесів, зниженню витрат на ветеринарне обслуговування та підвищенню продуктивності фермерського господарства. Загалом, така система створює надійну основу для сучасного управління здоров'ям тварин, забезпечуючи більш високий рівень добробуту корів і економічної ефективності ферми. Такий підхід забезпечує інтегровану і гнучку платформу, яка може адаптуватися до зростаючих потреб фермерського господарства і технологічних оновлень, підвищуючи ефективність управління здоров’ям тварин.

2.2 Розробка взаємодії програмних модулів системи моніторингу процесів життєдіяльності тваринницького складу

Розробка програмного модуля для моніторингу процесів життєдіяльності тваринницького складу фермерського господарства є комплексним завданням, яке включає інтеграцію з сенсорами, обробку даних, забезпечення їх безпеки та надання зручного інтерфейсу для користувачів. Такий модуль допоможе оптимізувати управління фермою, підвищити продуктивність тваринництва та покращити здоров'я тварин.

Для розробки програмного модуля для моніторингу процесів життєдіяльності тваринницького складу фермерського господарства найкраще застосувати об'єктно-орієнтоване програмування (ООП) з використанням модульного підходу. ООП та модульність забезпечать високу гнучкість, масштабованість та легкість у підтримці системи.

Інкапсуляція дозволяє групувати дані та методи, що працюють з цими даними, в єдині об'єкти. Це спрощує управління даними про тварин, зокрема їхніми фізіологічними показниками, станом здоров'я тощо. Наслідування, в свою чергу, сприятиме створенню ієрархій класів, що дозволяє легко розширювати систему, додаючи нові типи тварин або нові функціональні можливості [14]. Поліморфізм дозволяє обробляти різні типи об'єктів через спільний інтерфейс, що полегшує реалізацію універсальних алгоритмів для аналізу даних, моніторингу та діагностики. Повторне використання коду завдяки можливостям наслідування та модульності можна повторно використовувати вже написані класи та модулі, що знижує обсяг роботи та підвищує ефективність розробки [14].

Розподіл на модулі дозволяє розбивати систему на незалежні частини (модулі), які легко розробляти, тестувати та підтримувати. Для моніторингу стану показників здоров'я корів необхідно розробити кілька ключових модулів, кожен з яких виконує специфічні функції. Модульна розробка дозволить створити комплексну систему моніторингу стану здоров'я корів, яка забезпечить

своєчасне виявлення проблем та підвищить ефективність управління фермою, як вказано на рисунку 2.4.

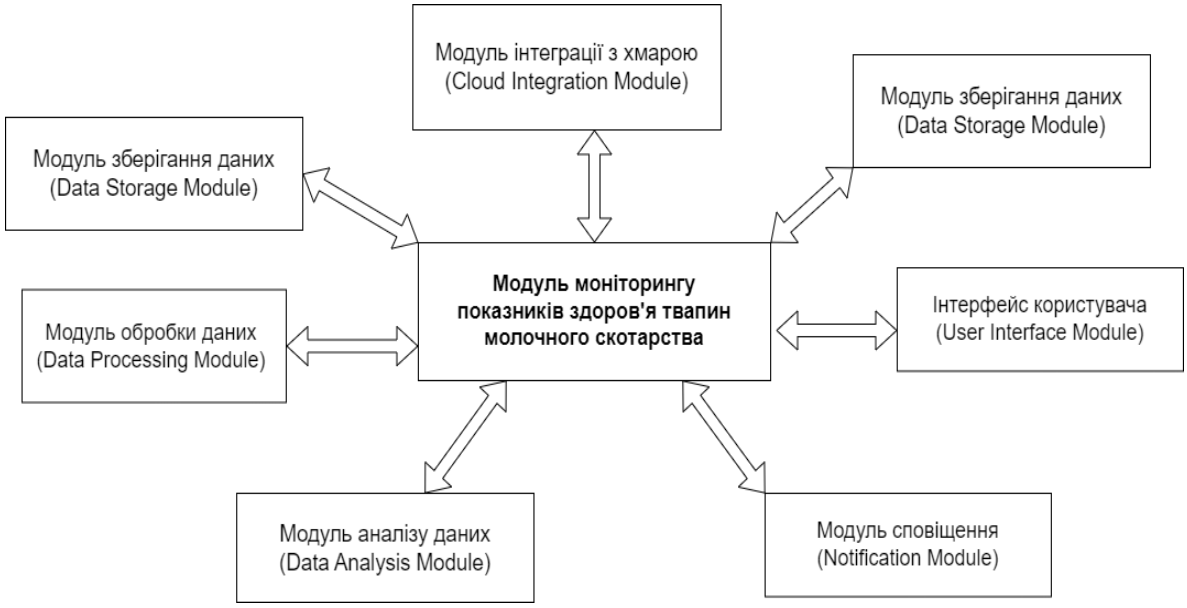

Рисунок 2.4 – Структурна схема програмного модуля для моніторингу процесів життєдіяльності тваринницького складу фермерського господарства

Згідно рисунку 2.4 розробимо більш детальний перелік модулів:

Модуль збору даних (Data Collection Module) забезбечуватиме інтеграцію з біосенсорами та включатиме такі показники здоров’я тварин молочного скотарства для моніторингу їх стану:

1) Температура тіла: Вимірювання внутрішньої температури для виявлення лихоманки або гіпотермії.
2) Рівень активності: Вимірювання рухової активності для виявлення періодів спокою та руху, що може свідчити про наявність хвороб або настання статевої охоти.
3) Пульс і серцевий ритм: Вимірювання частоти серцевих скорочень для оцінки загального стану здоров'я та виявлення стресу або хвороб.
4) Дихальна частота: Моніторинг частоти дихання для виявлення респіраторних проблем.

5) Споживання корму та води: Оцінка частоти та кількості прийомів корму і води для виявлення змін у харчовій поведінці, що можуть свідчити про хвороби або стрес.
6) Визначення настання статевої охоти, її циклу та тільності.
7) Рівень pH у рубці: Вимірювання рівня кислотності в рубці для виявлення проблем з травленням.
8) Інтенсивність жуйки: Вимірювання кількості та частоти жуйки для оцінки здоров'я травної системи.

Модуль обробки даних (Data Processing Module) міститиме аналіз та обробка сирих даних з датчиків для отримання корисної інформації. На основі цього імплемпентовані алгоритми виявлення аномалій для попередження виявлення відхилень від нормальних показників у тварини.

Модуль зберігання даних (Data Storage Module) включатиме базу даних забезпечить структуроване зберігання всіх зібраних даних.

Модуль аналізу даних (Data Analysis Module) відтворюватиме аналіз стану здоров'я допоможе у визначенні показників здоров'я корів на основі зібраних даних. Можливе подальше прогнозування проблем зі здоров'ям на основі отриманих даних з біосенсорів та графічне представлення показників здоров'я для зручності аналізу.

Модуль сповіщення (Notification Module) працюватиме система оповіщень, що допоможе працівникам фермерського господарства у разі виявлення аномалій або проблем зі здоров'ям. Сповіщення системи інформуватимуть фахівців через SMS або електронну пошту.

Інтерфейс користувача (User Interface Module) включатиме зручний крос-платформений застосунок, що забезпечуватиме доступ до даних для власників фермерських господарств за для відстеження коректної роботи працівників та врахування економічних показників та спеціалізованих фахівців. Включатиме основний віджет для управління всіма модулями та даними.

Модуль інтеграції з хмарою (Cloud Integration Module) забезпечуватиме передачу зібраних даних на хмарні сервери для подальшої обробки та зберігання.

Забезпечення доступу до даних з будь-якого пристрою та місця.

Модуль безпеки (Security Module) забезпечуватиме контроль доступу до даних і функцій системи для різних користувачів.

Таким чином для ефективного моніторингу процесів життєдіяльності тваринницького складу фермерського господарства необхідно створити комплексну систему, яка забезпечує збирання, аналіз і збереження даних про стан здоров'я корів. Ця система повинна включати кілька ключових компонентів: базу даних для зберігання інформації про кожну корову, модулі збору даних з різних датчиків, механізми аналізу та обробки даних, а також інтерфейси для користувачів.

База даних має містити інформацію про ідентифікатори корів, їх здоров'я, годівл, репродуктивну функцію та інші важливі параметри. Важливим аспектом є створення структури бази даних, яка дозволяє ефективно зберігати та оновлювати ці дані.

Подана інформація зберігається у базі даних, доступ до якої мають уповноважені особи, включаючи власників фермерських господарств та спеціалізованих фахівців, зокрема ветеринарних лікарів та зоотехніків. Це дозволяє проводити постійний моніторинг стану тварин і оперативно реагувати на будь-які відхилення від норми.

Крім того, дані можуть бути опубліковані у хмарі для забезпечення доступу до них з будь-якого місця, що сприяє підвищенню ефективності управління фермою. Використання такої системи дозволяє поліпшити стан здоров'я тварин, збільшити продуктивність ферми та знизити ризики виникнення захворювань шляхом своєчасного виявлення та реагування на потенційні проблеми.

Проєктування реляційної бази даних [15] є одним із найважливіших процесів для коректної роботи з даними. Подана база даних системи забезпечуватиме ранню діагностику хвороб корів на основі даних, отриманих із датчиків, потрібно розробити структуру, що включатиме таблиці з відповідними полями для зберігання показників здоров'я тварин, як на рисунку 2.5.

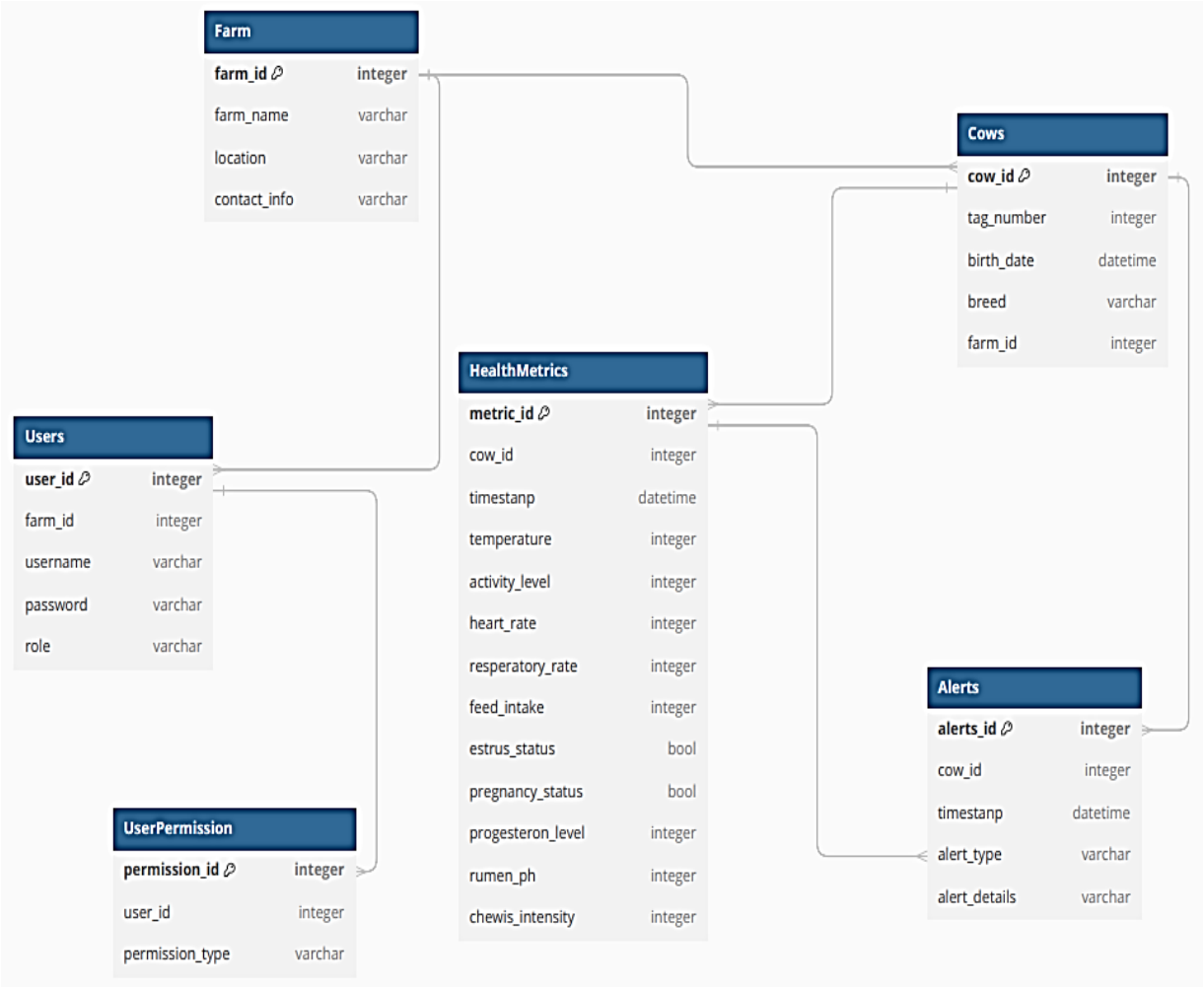

Рисунок 2.5 – UML-діаграма бази даних програмного модуля відстеження показників тварин

Розглянемо згідно рисунка 2.5 структуру бази даних більш детально:

1. Таблиця «Cows»:

- cow_id (унікальний ідентифікатор корови);
- tag_number (ідентифікаційний номер корови);
- birth_date (дата народження корови);
- breed (порода корови);
- farm_id (ідентифікатор ферми, де утримується корова);

2. Таблиця «HealthMetrics» (Показників здоров’я):

- metric_id (унікальний ідентифікатор показника);
- cow_id (ідентифікатор корови);

- timestamp (дата і час зняття показника);
- temperature (температура тіла);
- activity_level (рівень активності);
- heart_rate (пульс і серцевий ритм);
- respiratory_rate (дихальна частота);
- feed_intake (споживання корму);
- estrus_status (стан статевої охоти (true - в охоті, false - не в охоті));
- pregnancy_status (тільність (true - тільна, false - не тільна));
- progesterone_level (рівень прогестерону);
- rumen_ph (рівень pH у рубці);
- chewing_intensity (інтенсивність жуйки);

3. Таблиця «Alerts» (Сповіщення):

- alert_id (унікальний ідентифікатор сповіщення);
- cow_id (ідентифікатор корови);
- timestamp (дата і час створення сповіщення);
- alert_type (тип сповіщення);
- alert_details (деталі сповіщення);

4. Таблиця «Farm»:

- farm_id (унікальний ідентифікатор ферми);
- farm_name (назва ферми);
- location (розташування ферми);
- contact_info (контактна інформація);

5. Таблиця «Users»:

- user_id (унікальний ідентифікатор користувача);
- username (логін користувача);
- password (пароль користувача);
- role (роль користувача (наприклад, «фермер», «ветеринар», «менеджер»);
- farm_id (ідентифікатор ферми, до якої належить користувач);

6. Таблиця «UserPermissions»:

- permission_id (унікальний ідентифікатор дозволу);

- user_id (ідентифікатор користувача);

Ця структура бази даних дозволить ефективно збирати, зберігати та аналізувати дані про здоров'я корів, забезпечуючи своєчасне виявлення хвороб і інших проблем. Використання цих даних дозволить фермерам та ветеринарам приймати обґрунтовані рішення для покращення добробуту тварин та підвищення продуктивності господарства.

Важливо відзначити, що інтерфейс такого застосунку для моніторингу стану здоров'я корів має бути інтуїтивно зрозумілим, зручним та функціональним. На головному екрані повинна бути панель з основними показниками здоров'я корів, такими як температура, пульс, активність та споживання води та кормів, відображеними у вигляді графіків та діаграм. Інтерфейс повинен мати розділи для детального перегляду інформації про кожну корову, де можна побачити історію даних, аналіз її стану в реальному часі з допомогою датчиків та прогнози. Крім того, потрібен розділ для налаштувань, де користувач може налаштувати сповіщення та доступ до даних. Важливо також забезпечити можливість швидкого доступу до контактів ветеринарів та інших спеціалістів. Інтерфейс має бути адаптивним до різних розмірів екрану, а навігація повинна бути простою і логічною, щоб користувачі могли швидко знайти потрібну інформацію та приймати оперативні рішення на основі отриманих даних.

Ця інформація зберігається в базі даних, доступ до якої мають власники фермерських господарств і спеціалізовані фахівці, що дозволяє проводити постійний моніторинг стану тварин і оперативно реагувати на відхилення від норми. Дані можуть бути опубліковані в хмарі для забезпечення доступу з будь-якого місця, що підвищує ефективність управління фермою. Використання такої системи сприяє поліпшенню стану здоров'я тварин, збільшенню продуктивності ферми та зниженню ризиків виникнення захворювань шляхом своєчасного виявлення і реагування на потенційні проблеми.

2.3 Розробка алгоритму моніторингу процесів життєдіяльності тваринницького складу фермерського господарства

Розробка методів програмного модуля для моніторингу процесів життєдіяльності тваринницького складу фермерського господарства включає кілька ключових етапів і компонентів. Перш за все, це визначення цілей і вимог до системи, зокрема які саме показники здоров'я корів потрібно моніторити. Збір даних забезпечується інтеграцією з різними біосенсорами, такими як датчики температури, пульсу, активності, дихальної частоти, споживання корму та води, а також датчики для визначення pH у рубці та інтенсивності жуйки.

Важливим аспектом є забезпечення безперервного моніторингу і своєчасного сповіщення фермерів та ветеринарів про будь-які відхилення від норми. Зберігання даних організовується в базі даних, яка повинна бути структурованою і забезпечувати швидкий доступ до інформації. База даних має включати історію хвороб кожної корови, а також інформацію про її поточний стан здоров'я. Інтерфейс користувача, зокрема мобільний застосунок, повинен бути інтуїтивно зрозумілим і функціональним, надаючи доступ до всіх зібраних даних у вигляді графіків і діаграм. Він повинен дозволяти користувачам налаштовувати сповіщення і швидко отримувати доступ до контактів ветеринарів.

Алгоритми виявлення захворювань у корів працюють на основі аналізу даних, отриманих із різних датчиків. Для кожного типу захворювання існує окремий алгоритм. Алгоритм виявлення захворювань нервової системи починається зі збору даних про температуру тіла, рівень активності, пульс, серцевий ритм та інтенсивність жуйки. Загалом важливим етапом є загальна розробка алгоритмів моніторингу процесів життєдіяльності тваринницького складу фермерського господарства, який вказаний на рисунку 2.6.

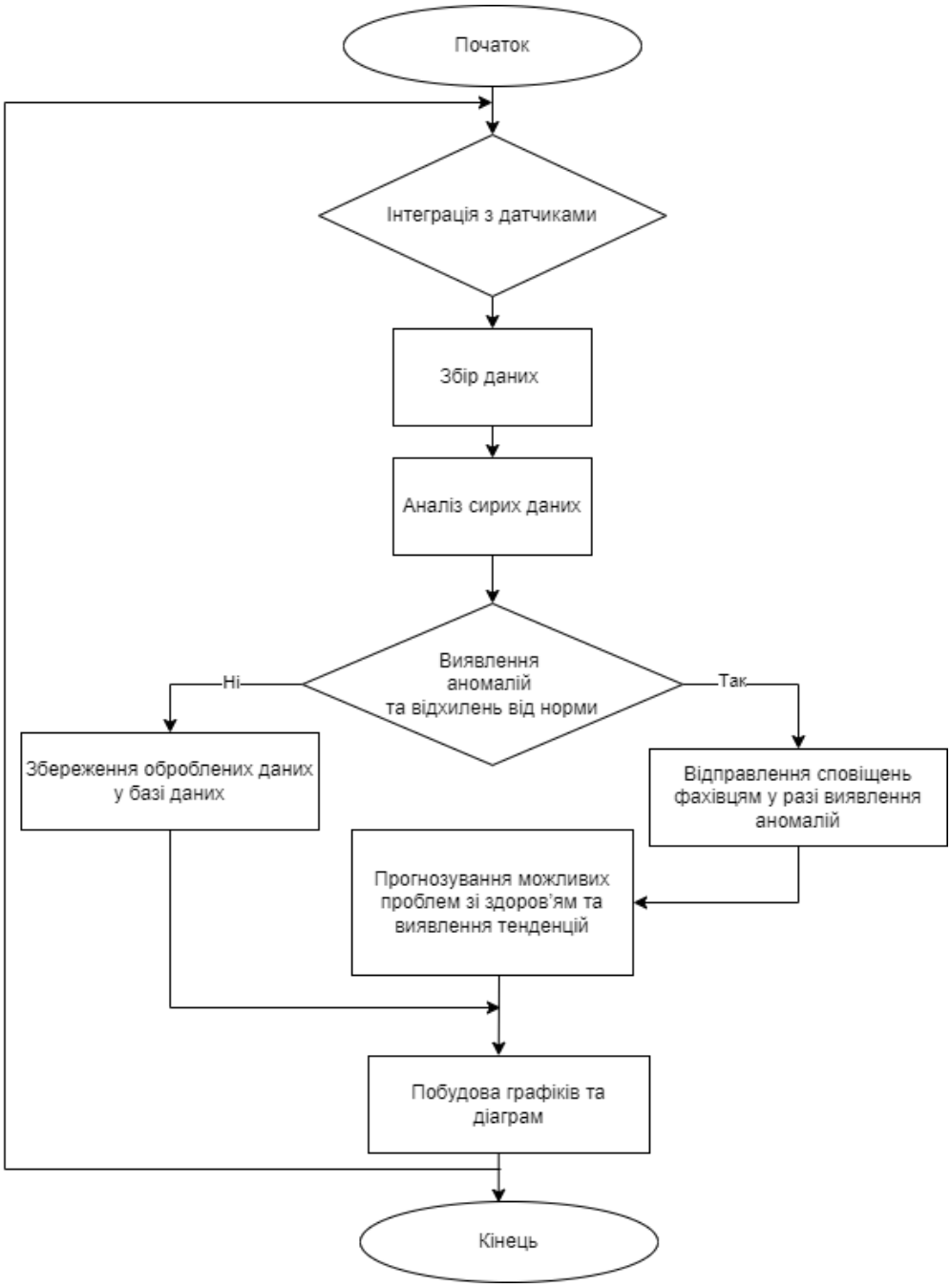

Рисунок 2.6 – Загальний алгоритм роботи програмного модуля моніторингу здоров’я тварин

Алгоритм виявлення інфекційних захворювань аналізує ті ж показники, зосереджуючись на температурі тіла та частоті дихання як основних критеріях, а також враховуючи споживання корму і води як додаткові критерії. Для виявлення паразитарних хвороб основні критерії включають аналіз температури та рівня активності, додаткові критерії можуть включати оцінку рівня pH у рубці.

Зобразимо алгоритм виявлення відхилень в процесі моніторингу стану здоров’я тварин зображено на рисунку 2.7.

Рисунок 2.7 – Принципова схема алгоритмів виявлення захворювань: а) – Алгоритм виявлення захворювань нервової системи; б) – алгоритм виявлення інфекційних захворювань; с) – алгоритм виявлення паразитарних хвороб; сині блоки позначають основні критерії алгоритмів; d) – зелені блоки вказують на додаткові критерії

Основні критерії включають перевірку температури тіла на аномальні

значення, аналіз рівня активності на відхилення та перевірку пульсу на підвищення або нерегулярності. Додаткові критерії включають оцінку інтенсивності жуйки, оскільки зниження цього показника може свідчити про нервові розлади. Алгоритм завершується після узгодження всіх критеріїв.

Зібрані дані з датчиків активності, температури, пульсу та інших біометричних параметрів передаються до центрального процесора для аналізу. На основі цих даних система може автоматично визначати ймовірність настання охоти, наближення отелення та можливі захворювання. Для виявлення охоти використовується аналіз моторної активності та температури тіла тварини. Якщо активність перевищує встановлений поріг, а температура підвищена, система фіксує стан охоти. Дані про репродуктивний статус також враховуються для точнішої діагностики. Приклад моніторингу даних з допомогою візуалізаціїї на рисунку 2.8.

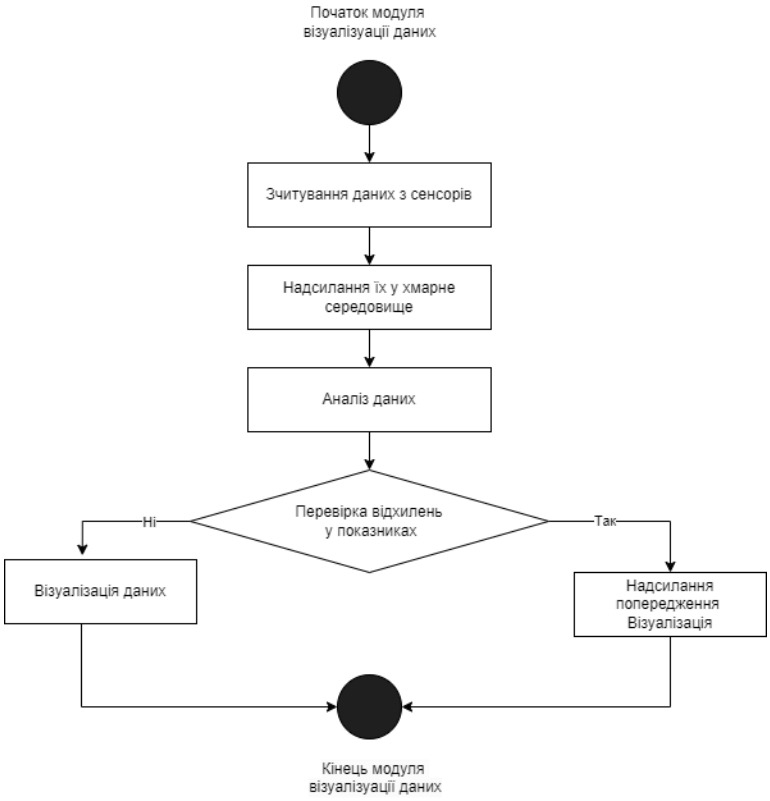

Рисунок 2.8 – Схема алгоритму моніторингу та візуалізації показників здоров’я тварин

Для визначення наближення отелення аналізується температура рубця і активність тварини. Якщо температура нижча за норму, а активність значно змінена, додатково перевіряється кількість актів пиття. Зниження споживання води свідчить про наближення отелення.

Загалом, розробка методів програмного модулю для моніторингу процесів життєдіяльності тваринницького складу фермерського господарства є важливим кроком у впровадженні сучасних технологій в аграрний сектор. Цей процес передбачає створення комплексної системи, яка забезпечує збір, зберігання, аналіз та візуалізацію даних про стан тварин. Дані з датчиків активності, температури, пульсу та інших біометричних параметрів передаються до центрального процесора, де вони аналізуються для визначення ймовірності настання охоти, наближення отелення та можливих захворювань.

2.3 Висновок до розділу 2

Розробка та проєктування програмного модуля для моніторингу показників здоров'я тварин на фермерському господарстві є ключовим етапом в сучасному аграрному секторі. Цей процес відображає поєднання передових технологій з традиційним методом господарювання, сприяючи підвищенню ефективності та стабільності в сільському господарстві. Основним завданням розробки є створення комплексної системи, що забезпечує збір, аналіз та відображення даних про стан здоров'я тварин.

Проєктування програмного модуля включає в себе розробку спеціалізованих модулів для збору та обробки даних, а також створення інтерфейсу для зручного використання користувачами. Результатом такої розробки є підвищення рівня моніторингу та контролю за здоров'ям тварин, що сприяє збільшенню продуктивності та ефективності фермерського комплексу.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЮ ДЛЯ МОНІТОРИНГУ ПРОЦЕСІВ ЖИТТЄДІЯЛЬНОСТІ ТВАРИН ФЕРМЕРСЬКОГО ГОСПОДАРСТВА

3.1 Обґрунтування вибору мов програмування та фреймворків

Вибір мов програмування для розробки мобільного додатку є критично важливим кроком, який визначає ефективність розробки, якість кінцевого продукту, його продуктивність та здатність задовольняти потреби користувачів.

Для розробки мобільних додатків під Android можна використовувати кілька різних мов програмування, кожна з яких має свої переваги та недоліки. Java є офіційною мовою програмування для Android з моменту його запуску. Вона має велику кількість ресурсів, документації та прикладів коду, а також величезну спільноту розробників, що забезпечує підтримку та допомогу [15].

Java достатньо продуктивна, вона поступається C++ та Kotlin у деяких сценаріях.Kotlin є сучасною мовою, розробленою для вирішення багатьох недоліків Java. Вона є офіційно підтримуваною мовою для Android розробки. Kotlin дозволяє писати менше коду завдяки сучасним конструкціям та синтаксичним цукрам, а також має покращену безпеку щодо нульових значень, що зменшує кількість потенційних помилок. Легка інтеграція з існуючим Java-кодом також є перевагою. Однак спільнота Kotlin все ще менша порівняно з Java, і можуть виникати проблеми з пошуком конкретних прикладів та рішень через відносну новизну мови [16].

C++ забезпечує високу продуктивність завдяки нативному коду, що є критичним для обчислювально інтенсивних додатків. Більший контроль над пам'яттю та системними ресурсами є значною перевагою. Проте C++ має високий рівень складності у порівнянні з іншими мовами, більш складний синтаксис та вимоги до управління пам'яттю. Крім того, кількість ресурсів та прикладів менша порівняно з Java та Kotlin [16].

Dart з Flutter надає можливість створення додатків одразу для Android та iOS з єдиною базою коду, що є великою перевагою для кросплатформенних розробок. Висока продуктивність досягається завдяки компіляції в нативний код, а функція "Hot Reload" дозволяє швидко бачити зміни у коді, що прискорює цикл розробки. Flutter має багату бібліотеку готових віджетів для створення інтерфейсу користувача. Проте Flutter є відносно новою технологією, яка ще не є такою перевіреною часом, як Java, і потребує вивчення нової мови програмування Dart та фреймворку Flutter [17].

Вибір Flutter та Dart для розробки додатку для відстежування здоров'я тварин було обґрунтованим рішенням через кілька важливих причин, які відповідають специфічним вимогам та цілям цього проекту. Нижче опишемо основні переваги

1) Кросплатформенність: Flutter дозволяє створювати додатки для кількох платформ, зокрема Android та iOS, використовуючи єдину базу коду. Це означає, що один і той самий додаток може працювати на різних мобільних пристроях без додаткових зусиль для окремої розробки, що знижує витрати та час на розробку і підтримку.

2) Висока продуктивність: Завдяки тому, що Dart, мова програмування, яка використовується у Flutter, компілюється в нативний код, додатки на Flutter мають високу продуктивність і швидкодію, що є критично важливим для додатків, які займаються моніторингом і обробкою даних у реальному часі.

3) Швидкий розвиток та "Hot Reload": Функція "Hot Reload" дозволяє розробникам миттєво бачити зміни в коді без необхідності перезапускати додаток [18]. Це значно прискорює процес розробки, тестування і налагодження, що дозволяє швидше впроваджувати нові функції та виправляти помилки.

4) Багатий набір віджетів: Flutter надає розробникам широкий вибір готових віджетів для створення красивих та інтуїтивних інтерфейсів користувача. Це дозволяє створити зручний і привабливий інтерфейс додатку для фермерів та ветеринарів, що робить процес моніторингу здоров'я тварин простішим і ефективнішим.

5) Стабільність і надійність: Завдяки компіляції у нативний код, додатки на Flutter відзначаються стабільністю і високою продуктивністю. Це особливо важливо для додатків, які потребують надійної роботи і здатності обробляти великі обсяги даних у реальному часі, таких як додатки для моніторингу здоров'я тварин.

6) Гнучкість та адаптивність: Flutter дозволяє легко адаптувати інтерфейс додатку під різні розміри екранів та пристроїв, що є важливим для забезпечення доступності та зручності використання додатку на різних типах пристроїв, які можуть використовувати фермери та ветеринари.

Попри те, що Flutter та Dart має багато переваг, він також має певні недоліки, які варто враховувати при виборі технології для розробки додатку для відстежування здоров'я тварин на фермерському господарстві. Розмір додатку є одним з основних недоліків Flutter є те, що додатки, створені з його допомогою, зазвичай мають більший розмір у порівнянні з нативними додатками. Це може бути проблемою, особливо в умовах обмеженого інтернет-з'єднання або для користувачів з пристроями, які мають обмежену пам'ять. Flutter надає багато вбудованих віджетів і можливостей, але можуть виникати ситуації, коли доступ до специфічних нативних функцій або інтеграція з певними сторонніми бібліотеками є складнішою.

Для проектування додатку з моніторингу показників здоров'я тварин на фермерському господарстві використовується SQL, оскільки ця мова забезпечує ефективну роботу з реляційними базами даних, що є критичним для обробки великих обсягів даних про тварин. SQL (Structured Query Language) пропонує потужні можливості для виконання складних запитів, що дозволяє витягувати необхідну інформацію для аналізу стану здоров'я тварин у режимі реального часу [19].

Однією з ключових переваг SQL є можливість забезпечення цілісності даних завдяки використанню транзакцій та обмежень, що дозволяє уникнути дублювання і втрати даних. Це особливо важливо для системи моніторингу, де точність і повнота інформації мають вирішальне значення для прийняття

оперативних рішень. Крім того, SQL підтримує масштабованість, що дозволяє легко розширювати базу даних у міру збільшення кількості тварин і обсягів даних, що збираються.

SQL також добре інтегрується з іншими технологіями, використовуваними у розробці додатків. Використання Flutter для створення інтерфейсу користувача дозволяє розробляти кросплатформенні додатки, які працюють на різних операційних системах, таких як Android і iOS, з одного коду [19]. Dart, мова програмування, на якій базується Flutter, забезпечує високу продуктивність та легкість у написанні логіки додатку. Поєднання цих технологій забезпечує зручність розробки, швидкодію додатку та його високу якість.

Використання розробки на основі Dart, Flutter та SQL має безліч переваг, які сприяють швидкому і ефективному створенню мобільних додатків. Dart - це сучасна та потужна мова програмування з простим синтаксисом, яка дозволяє розробникам швидко писати структурований і легко читабельний код. Flutter, у свою чергу, є потужним SDK для створення красивих та високопродуктивних інтерфейсів користувача, який забезпечує гнучкість та швидкість розробки. SQL - це мова запитів, яка дозволяє ефективно управляти базами даних та забезпечує доступ до різноманітних даних [19].

Використання цих технологій у поєднанні дозволяє створювати мобільні додатки з високою продуктивністю, швидкістю та надійністю. Dart та Flutter забезпечують можливість швидко розробляти кросплатформенні додатки з одним кодом, що значно заощаджує час та зусилля розробника. SQL, у свою чергу, дозволяє ефективно управляти базами даних та забезпечує потужні можливості для зберігання та обробки інформації.

Загалом, використання Dart, Flutter та SQL для розробки додатку для моніторингу здоров'я тварин на фермерському господарстві є ефективним та перспективним підходом. Ці технології забезпечують швидку розробку, кросплатформенність, гнучкість у розробці інтерфейсу користувача та надійність у роботі з базами даних. Вони дозволяють створювати зручні та функціональні додатки, які допомагають фермерам ефективно відстежувати та

доглядати за здоров’ям тварин.

3.2 Програмна реалізація модулю для моніторингу процесів життєдіяльності тваринницького складу

Програмна реалізація модуля для моніторингу процесів життєдіяльності тваринницького складу, яка використовує архітектурний підхід MVC (Model-View-Controller), принципи SOLID, та технології Flutter і Dart, забезпечує ефективне і надійне рішення для відстеження показників здоров'я тварин [21].

Model (Модель): Модель відповідає за управління даними та бізнес-логікою додатку. У випадку моніторингу здоров'я тварин, модель включає структури для зберігання даних про тварин, такі як температура, пульс, рівень активності, споживання корму та води, дихальна частота та інші важливі параметри. Використання SQL забезпечує ефективне зберігання та управління цими даними, дозволяючи швидко доступати до необхідної інформації.

View (Представлення): Представлення відповідає за відображення даних та взаємодію з користувачем. Використання Flutter дозволяє створити багатофункціональні та зручні у використанні інтерфейси користувача, які працюють на різних платформах iOS, Android. Завдяки Flutter можна швидко та легко розробити красивий і адаптивний інтерфейс для фермерів, що включає графіки, діаграми та інші візуальні елементи для відображення стану здоров'я тварин.

Controller (Контролер): Контролер відповідає за прийняття рішень на основі взаємодії між моделлю та представленням. Він обробляє запити користувача, отримує дані з моделі, та оновлює представлення відповідно до цих даних [21]. У випадку моніторингу здоров'я тварин, контролер може реалізовувати алгоритми аналізу даних та виявлення відхилень від нормальних показників, а також сповіщати користувачів про необхідність втручання. Розглянемо поданий MVC архітектурний шаблон на рисунку 3.1.

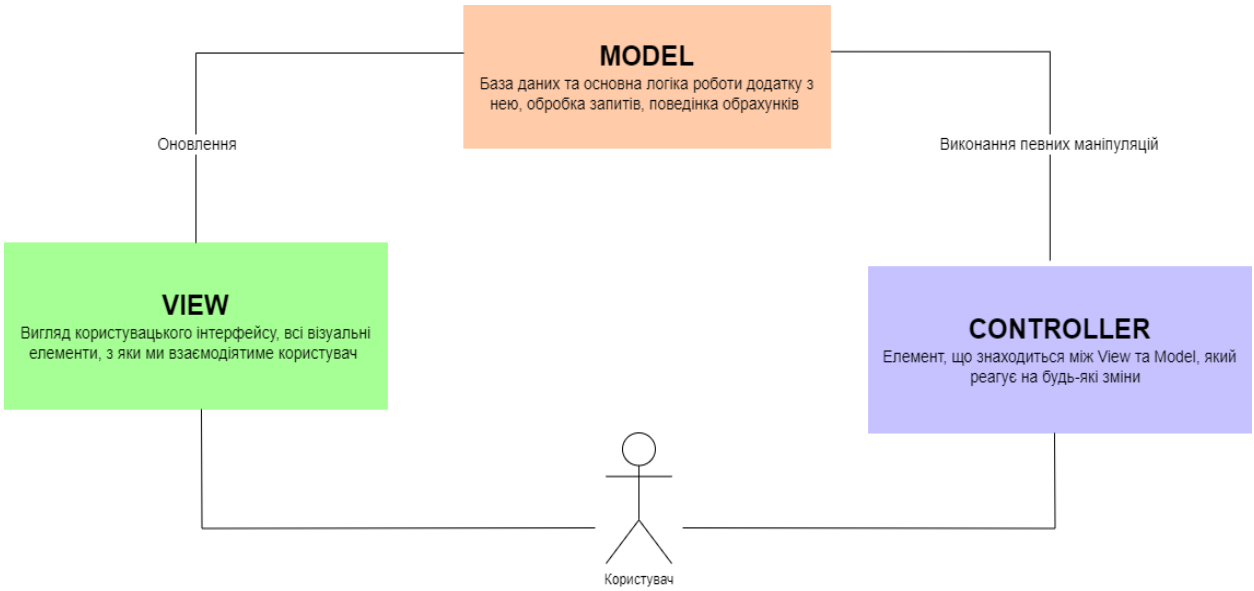

Рисунок 3.1 – Принципова схема роботи архітектурного шаблону MVC

Використання принципів SOLID в розробці програмного модуля для моніторингу показників здоров'я тварин на фермерському господарстві є дуже корисним з кількох причин.

Принцип єдиного обов'язку (Single Responsibility Principle) забезпечує, що кожен клас і модуль системи виконують одну чітко визначену задачу [20]. Це значно спрощує розробку, тестування та підтримку коду, оскільки зміни в одній частині системи не призводять до непередбачуваних наслідків в інших частинах. Наприклад, клас, відповідальний за зберігання даних про температуру тварин, не буде містити логіку для відображення цієї інформації на екрані.

Принцип відкритості/закритості (Open/Closed Principle) дозволяє легко розширювати функціональність системи без необхідності змінювати існуючий код. Це особливо важливо в умовах постійного розвитку системи, коли потрібно додавати нові функції або змінювати алгоритми аналізу даних. Наприклад, додавання нового типу сенсора або нового алгоритму обробки даних може бути реалізоване без зміни основних компонентів системи.

Принцип підстановки Лісков (Liskov Substitution Principle) гарантує, що об'єкти підкласів можуть заміняти об'єкти базових класів без порушення логіки програми. Це забезпечує коректну роботу поліморфізму і спрощує розширення

системи новими класами, що зберігають узгодженість з існуючими інтерфейсами [20].

Принцип розділення інтерфейсів (Interface Segregation Principle) сприяє створенню невеликих, конкретних інтерфейсів замість одного універсального. Це дозволяє уникнути ситуацій, коли класи змушені реалізовувати методи, які вони не використовують. Наприклад, інтерфейс для сенсора температури може бути різним від інтерфейсу для сенсора пульсу, навіть якщо обидва сенсори інтегруються в загальну систему.

Принцип інверсії залежностей (Dependency Inversion Principle) забезпечує, що модулі високого рівня не залежать від модулів низького рівня. Обидва типи модулів повинні залежати від абстракцій. В цілому, це дозволяє зменшити зв'язність коду і полегшує впровадження змін. Наприклад, обробка даних від сенсорів може бути реалізована за допомогою різних алгоритмів, які легко замінюються завдяки використанню абстракцій [20].

В першу чергу для виконання програмної реалізації модуля відстеження процесів життєдіяльності тварин потрібно виконати інтеграцію моільного застосунку із датчиками, які передаватимуть усю необхідну інформацію про здоров'я корів на фермерському господарстві з допомогою методу SensorServiceCow через Wi-Fi модуль:

```
class SensorServiceCow {
  final String baseUrl;
  SensorService({required this.baseUrl});
  // Метод для отримання показників здоров'я з датчиків за
допомогою HTTP протоколу
  Stream<List<HealthMetric>> getHealthMetrics() async* {
    // Створюємо URL для запиту до сервера
    final url = Uri.parse('$baseUrl/health_metrics');
    // Виконуємо запит GET до сервера
    final response = await http.get(url);
    // Перевіряємо статус відповіді
    if (response.statusCode == 200) {
      // Розпаковуємо відповідь JSON
      final List<dynamic> jsonData = jsonDecode(response.body);
      // Конвертуємо JSON дані в об'єкти HealthMetric
      final List<HealthMetric> metrics =
          jsonData.map((data)                                    =>
HealthMetric.fromMap(data)).toList();
      // Повертаємо список показників здоров'я через потік
```

```
      yield metrics;
    } else {
      // Якщо відповідь не 200 (наприклад, 404 або 500), генеруємо
виключення
      throw Exception('Failed to fetch health metrics');
    }
  }
class SensorService {
  // Приклад методу для отримання показників здоров'я з датчиків
  Stream<List<HealthMetric>> getHealthMetrics() async* {
    // Симулюємо отримання даних кожну секунду протягом 1 години
    for (int i = 0; i < 216000; i++) {
            List<HealthMetric> metrics = [
        HealthMetric(
          metricId: '1',
          cowId: 'cow1',
          timestamp: DateTime.now(),
          temperature: 38.5 + (i % 3),
          activityLevel: 80 + (i % 5),
          heartRate: 70 + (i % 10),
          respiratoryRate: 20 + (i % 5),
          feedIntake: 5.5 + (i % 2),
          estrusStatus: i.isOdd,
          pregnancyStatus: i.isEven,
          progesteroneLevel: 2.5 + (i % 3),
          rumenPh: 6.8 + (i % 2),
          chewingIntensity: 100 - (i % 20),
        ),
 ];}.
```

Також перед реалізацією головного модуля відстеження та виявлення показників від норми у здоров’ї корів маємо метод додавання, видалення та редагування даних про тварин на фермерському господарстві. Для роботи з базою даних у Flutter ви можете скористатися пакетом sqflite для локальної SQLite бази даних [19]. Ось невеликий приклад коду для додавання, видалення та редагування даних про тварин у програмі:

```
class Cow {
       int id;
       String tagNumber;
       DateTime birthDate;
       String breed;
       int farmId;
       Cow({this.id,   this.tagNumber,   this.birthDate,   this.breed,
     this.farmId});
```

```
  Map<String, dynamic> toMap() {
    return {
      'id': id,
      'tagNumber': tagNumber,
      'birthDate': birthDate.toIso8601String(),
      'breed': breed,
      'farmId': farmId,
    };
  static Cow fromMap(Map<String, dynamic> map) {
    return Cow(
      id: map['id'],
      tagNumber: map['tagNumber'],
      birthDate: DateTime.parse(map['birthDate']),
      breed: map['breed'],
      farmId: map['farmId'],
    );}
class DatabaseHelper {
  static       final       DatabaseHelper       _instance       =
DatabaseHelper._internal();
  factory DatabaseHelper() => _instance;
  static Database _database;
  DatabaseHelper._internal();
  Future<Database> get database async {
    if (_database != null) return _database;
    _database = await initDatabase();
    return _database;
  }
  Future<Database> initDatabase() async {
    final path = await getDatabasesPath();
    return await openDatabase(databasePath, version: 1, onCreate:
(db, version) async {
      await db.execute('''
        CREATE TABLE cows(
          id INTEGER PRIMARY KEY AUTOINCREMENT,
          tagNumber TEXT,
          birthDate TEXT,
          breed TEXT,
          farmId INTEGER
        )
      ''');});};}
  Future<void> insertCow(Cow cow) async {
    final db = await database;
    await db.insert('cows', cow.toMap());
  }
  Future<void> updateCow(Cow cow) async {
    final db = await database;
    await  db.update('cows',  cow.toMap(),  where:  'id  =  ?',
whereArgs: [cow.id]);
  }
  Future<void> deleteCow(int id) async {
    final db = await database;
```

```
    await db.delete('cows', where: 'id = ?', whereArgs: [id]);
  }.
```

У цьому коді ми створили можливість роботи з базою даних у Flutter, використовуючи пакет sqflite, що дозволяє працювати з локальною SQLite базою даних. Спочатку ми визначили модель даних для корів за допомогою класу Cow, який містить необхідні поля і методи для конвертації даних у формат, зрозумілий базі даних.

Потім ми створили клас DatabaseHelper, який є допоміжним і містить методи для ініціалізації бази даних, а також для вставки, оновлення та видалення записів про корів. Цей клас дозволяє спростити роботу з базою даних та забезпечує можливість виконувати всі необхідні операції з даними про корів у застосунку. Зокрема, методи insertCow, updateCow та deleteCow дозволяють додавати нових корів, оновлювати існуючі дані та видаляти записи відповідно. Це дає розробнику зручний і простий інтерфейс для керування даними про корів у застосунку.

Дуже важливим є загальний модуль для виявлення хвороби та відправлення повідомлень про попередження ветеринарів про відхилення у показниках здоров'я корів, використовуючи вказані параметри, де CowHealthMonitor використовується для моніторингу здоров'я корів. HealthAnalyzer проводить аналіз показників здоров'я корів і виявляє відхилення. HealthAlertNotifier відправляє повідомлення про попередження ветеринарів, якщо виявлені відхилення. HealthMetrics - це клас, який представляє показники здоров'я корів, які аналізуються:

```
class HealthAnalyzer {
  final HealthAlertNotifier alertNotifier;
  HealthAnalyzer({required this.alertNotifier});
  void analyzeHealth(HealthMetrics metrics) {
    // Проведення аналізу показників здоров'я
    if (metrics.temperature > 39.0) {
      // Висока температура, відправити попередження
      alertNotifier.sendAlert(
        veterinarianEmail: 'veterinarian@example.com',
        cowId: metrics.cowId,
        message:  'Попередження:  Висока  температура  корови
${metrics.cowId} (${metrics.temperature}°C)',
      );}
```

```
    if (metrics.activityLevel < 5.0) {
      // Низький рівень активності, відправити попередження
      alertNotifier.sendAlert(
        veterinarianEmail: 'veterinarian@example.com',
        cowId: metrics.cowId,
        message: 'Попередження: Низький рівень активності у корови
${metrics.cowId} (${metrics.activityLevel})',
      );}
    if (metrics.heartRate > 100.0) {
      // Підвищений пульс, відправити попередження
      alertNotifier.sendAlert(
        veterinarianEmail: 'veterinarian@example.com',
        cowId: metrics.cowId,
        message:  'Попередження:  Підвищений  пульс  у  корови
${metrics.cowId} (${metrics.heartRate} bpm)',
      );}
    if (metrics.respiratoryRate > 25) {
      // Підвищена частота дихання, відправити попередження
      alertNotifier.sendAlert(
        veterinarianEmail: 'veterinarian@example.com',
        cowId: metrics.cowId,
        message: 'Попередження: Підвищена частота дихання у корови
${metrics.cowId} (${metrics.respiratoryRate} bpm)',
      );}
    if (metrics.feedIntake < 10.0) {
      // Низьке споживання корму, відправити попередження
      alertNotifier.sendAlert(
        veterinarianEmail: 'veterinarian@example.com',
        cowId: metrics.cowId,
        message: 'Попередження: Низьке споживання корму у корови
${metrics.cowId} (${metrics.feedIntake} кг)',
      ); }
    if (metrics.progesteroneLevel < 2.0) {
      // Низький рівень прогестерону, відправити попередження
      alertNotifier.sendAlert(
        veterinarianEmail: 'veterinarian@example.com',
        cowId: metrics.cowId,
        message:  'Попередження:  Низький  рівень  прогестерону  у
корови ${metrics.cowId} (${metrics.progesteroneLevel} ng/mL)',
      ); }
    if (metrics.rumenPh < 6.0 || metrics.rumenPh > 7.0) {
      // Відхилення рівня pH у рубці, відправити попередження
      alertNotifier.sendAlert(
        veterinarianEmail: 'veterinarian@example.com',
        cowId: metrics.cowId,
        message:  'Попередження:  Відхилення  рівня  pH  у  рубці  у
корови ${metrics.cowId} (${metrics.rumenPh})',
      ); }
    if (metrics.chewingIntensity < 50.0) {
      // Низька інтенсивність жуйки, відправити попередження
      alertNotifier.sendAlert(
```

```
        veterinarianEmail: 'veterinarian@example.com',
        cowId: metrics.cowId,
        message: 'Попередження: Низька інтенсивність жуйки у
корови ${metrics.cowId} (${metrics.chewingIntensity}%)',
      ); }
HealthAlertNotifier {
  final String username;
  final String password;
  HealthAlertNotifier({required     this.username,     required
this.password;
}.
```

Реалізація візуалізації показників здоров'я тварин на фермі є критично важливою для ефективного та успішного ведення сільського господарства з кількох причин. Візуалізація дозволяє фермерам та ветеринарам легко спостерігати за змінами у стані здоров'я тварин. Графічне відображення показників, таких як температура тіла, активність, пульс, частота дихання та інші, дозволяє швидко виявляти відхилення в цих показниках, що може свідчити про потенційні проблеми зі здоров'ям тварин.

Найголовніше це допомагає в аналізі та порівнянні даних з різних періодів часу. Це дозволяє фермерам виявляти тенденції та зміни у здоров'ї тварин, а також ефективно вживати заходів для їх покращення. Наприклад, виявлення зменшення виробництва молока у певний період часу може вказувати на можливі проблеми зі здоров'ям корів або на необхідність змін у годівлі. Згідно цього матимемо таку реалізацію:

```
class HealthChart extends StatelessWidget {
  final List<double> data;
  final List<String> labels;

  HealthChart({required this.data, required this.labels});
  @override

  Widget build(BuildContext context) {
    return Container(
      height: 300,
      child: LineChart(
              spots: List.generate(
                data.length,
                (index) => FlSpot(index.toDouble(), data[index]),
                          leftTitles: SideTitles(
              showTitles: true,
              getTextStyles:   (context,   value)   =>   const
```

```
TextStyle(color: Colors.black),
              margin: 8,
              reservedSize: 28,
              interval: 5,
            ),),
          borderData:    FlBorderData(show:    true,    border:
Border.all(color: Colors.black)),
          minX: 0,
value : element) + 5,),),);}
class HealthStatisticsChart extends StatelessWidget {
  final List<List<double>> data;
  @override
  Widget build(BuildContext context) {
    return Container(
      height: 300,
      child: LineChart(
        LineChartData(
          lineBarsData: List.generate(
            data.length,
            (index) => LineChartBarData(
              spots: List.generate(
                data[index].length,
                (subIndex)    =>    FlSpot(subIndex.toDouble(),
data[index][subIndex]),
                    borderData: FlBorderData(show: true, border:
Border.all(color: Colors.black)),
          minX: 0,
          maxX: (data.first.length - 1).toDouble().
```

Цей код містить два види графіків: HealthChart, який відображає показники здоров'я для одного періоду часу, та HealthStatisticsChart, який відображає загальну статистику для кількох періодів часу. Обидва графіки використовують LineChart з пакету fl_chart, щоб побудувати лінійну діаграму з даними [19].

3.3 Тестування програмного модулю моніторингу процесів життєдіяльності тварин на фермерському господарстві

При тестуванні програмного модулю моніторингу процесів життєдіяльності тварин на фермерському господарстві важливо переконатися, що всі його функції працюють правильно та надійно. Спочатку потрібно перевірити додавання нових корів у систему, впевнившись, що вони зберігаються коректно. Потім слід перевірити можливість оновлення та видалення інформації про корів, переконавшись, що зміни відображаються

правильно та зберігаються в базі даних.

Далі важливо перевірити відображення показників здоров'я кожної тварини на графіках та інших інтерфейсах програми. Також слід спостерігати за повідомленнями, які відображаються у випадку виявлення відхилень у показниках здоров'я тварин. Важливо перевірити взаємодію модулю з іншими функціями системи, щоб переконатися, що дані передаються правильно. Тестування модулю допоможе забезпечити його правильну роботу та виявити та виправити можливі недоліки чи проблеми у його функціональності.

Тестуваня мобільного додатку проводиться у редакторі коду Android Studio з допомогою емулятора Pixel 7 Pro, Android версія 13. Сервер мобільного застосунку, що оброблятиме запити за адресою http://localhost:3000/bdcows. На рисунку 3.2 відобразимо віджети спливаючого вікна, віджет входу та віджет основго вікна мобільного застосунку.

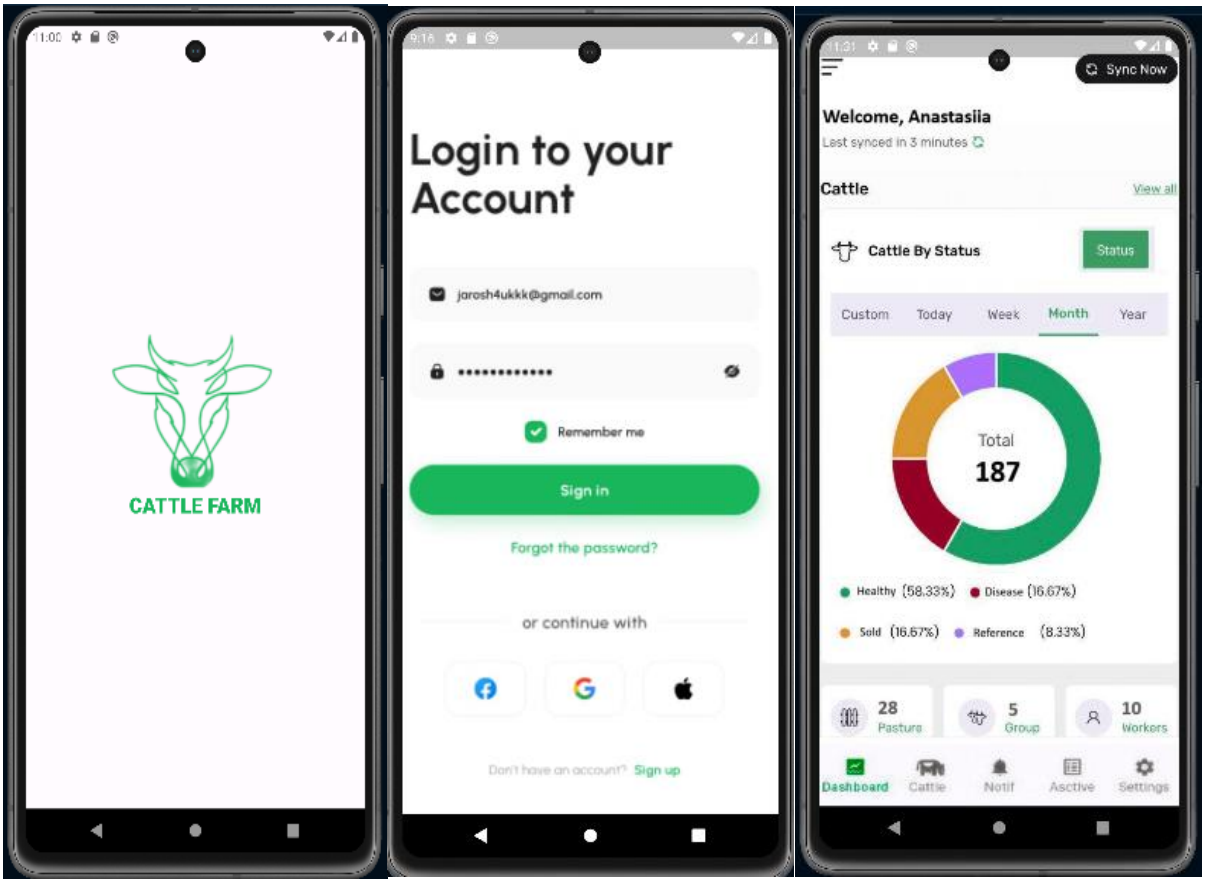

Рисунок 3.2 – Загальний вигляд користувацького інтерфейсу реалізованої програми CattleFarm

Для відображення екранів для реєстрації нових тварин у додатку створено інтуїтивний та зручний інтерфейс, який дозволить користувачам легко вводити інформацію про нових тварин. В меню знизу є можливість реєстрації матиме знапис "Add Animal". Під заголовком буде розміщено опис, що закликає користувачів заповнити наведену нижче інформацію, щоб зареєструвати нову тварину у фермерському господарстві. Ідентифікаційний номер тварини (Tag Number): Введіть унікальний ідентифікаційний номер тварини. Дата народження (Birth Date): Виберіть дату народження тварини з календаря. Порода (Breed): Введіть породу тварини. Ідентифікатор ферми (Farm ID): Введіть ідентифікатор ферми, де утримується тварина як на рисунку 3.3.

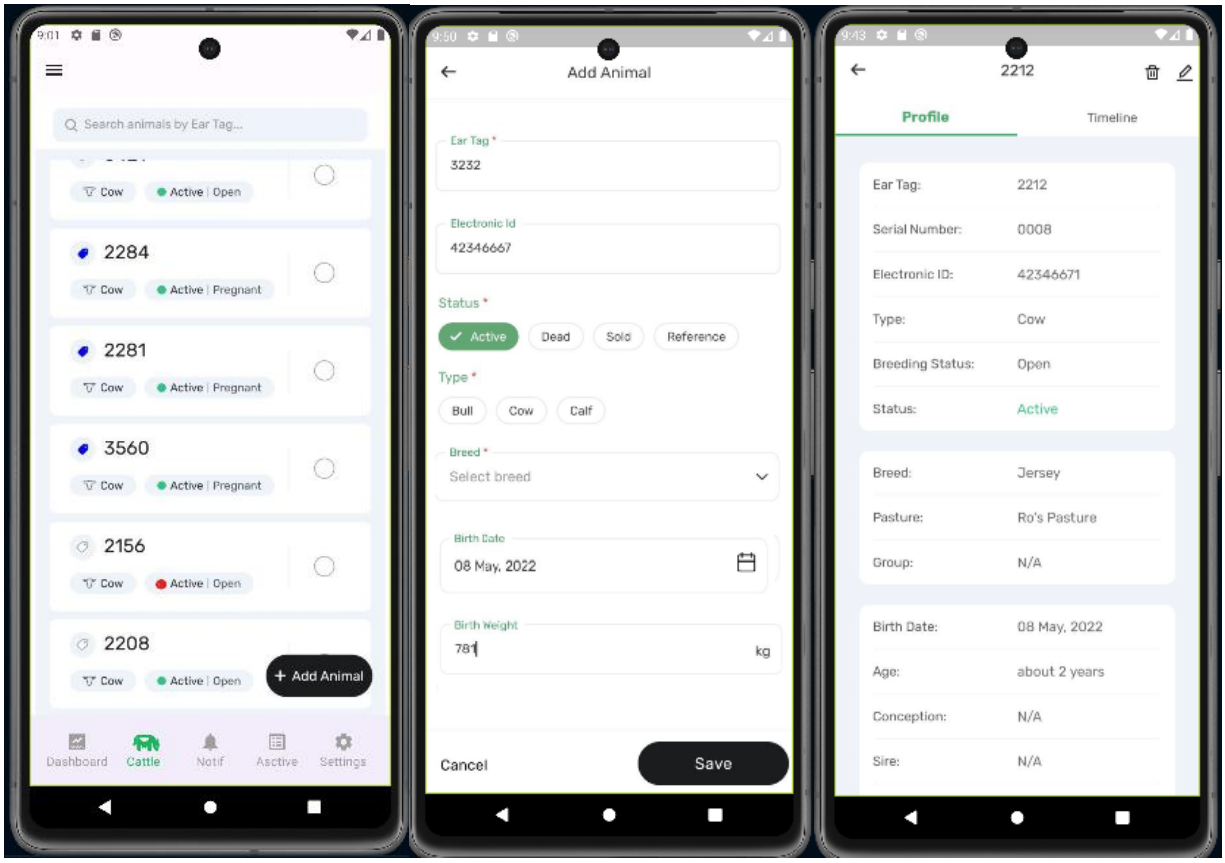

Рисунок 3.3 – Загальний вигляд інтерфейсного вікна введення обліку тварин

Тестування візуалізації даних показників здоров'я корів є важливим для забезпечення коректного відображення інформації, яка допомагає фермерам і

ветеринарам швидко оцінювати стан тварин. Для цього було виконано декілька ключових кроків. Спочатку підготували дані для тестування, використовуючи реальні або наближені до реальних дані, які включають нормальні випадки, граничні значення та помилкові дані для перевірки обробки винятків. Потім перевірено коректність відображення даних на графіках та інших візуальних компонентах. Переконалися, що всі показники, такі як температура, рівень активності, пульс, дихальна частота, споживання корму, стан статевої охоти, тільність, рівень прогестерону, рівень pH у рубці та інтенсивність жуйки, відображаються правильно як на рисунку 3.4.

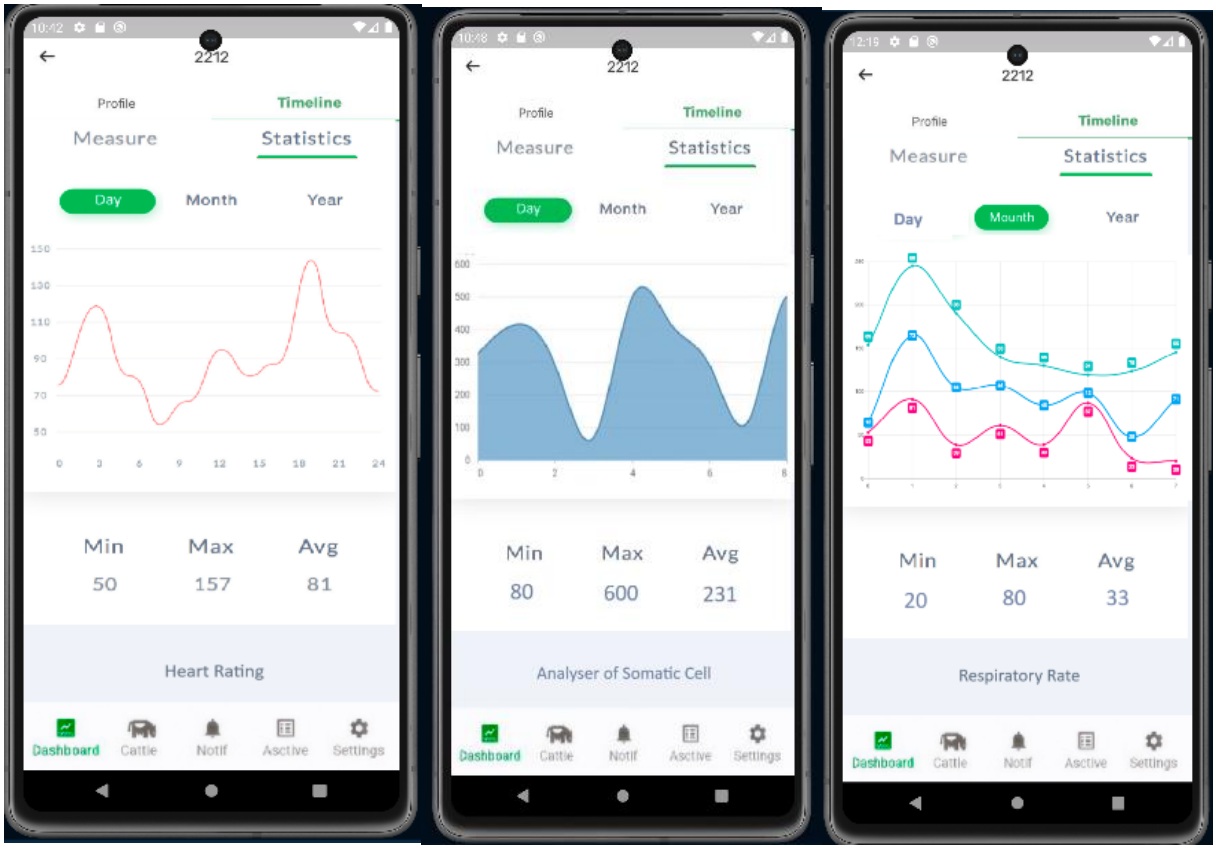

Рисунок 3.4 – Загальний вигляд віджетів моніторингу показників здоров’я тварин у вигляді графіків

Тестування сповіщень при відхиленні у показниках здоров'я корів полягає у перевірці генерації, доставки та відображення сповіщень користувачам в разі виявлення аномалій у показниках. Ключові аспекти включають перевірку коректності генерації сповіщень, їх вчасну доставку, чітке відображення та

можливість взаємодії з ними, а також тестування в реальному середовищі з реальними даними і пристроями. На рисунку 3.5 відображено інтерфейс сповіщень та можливість ведення цієї тварини.

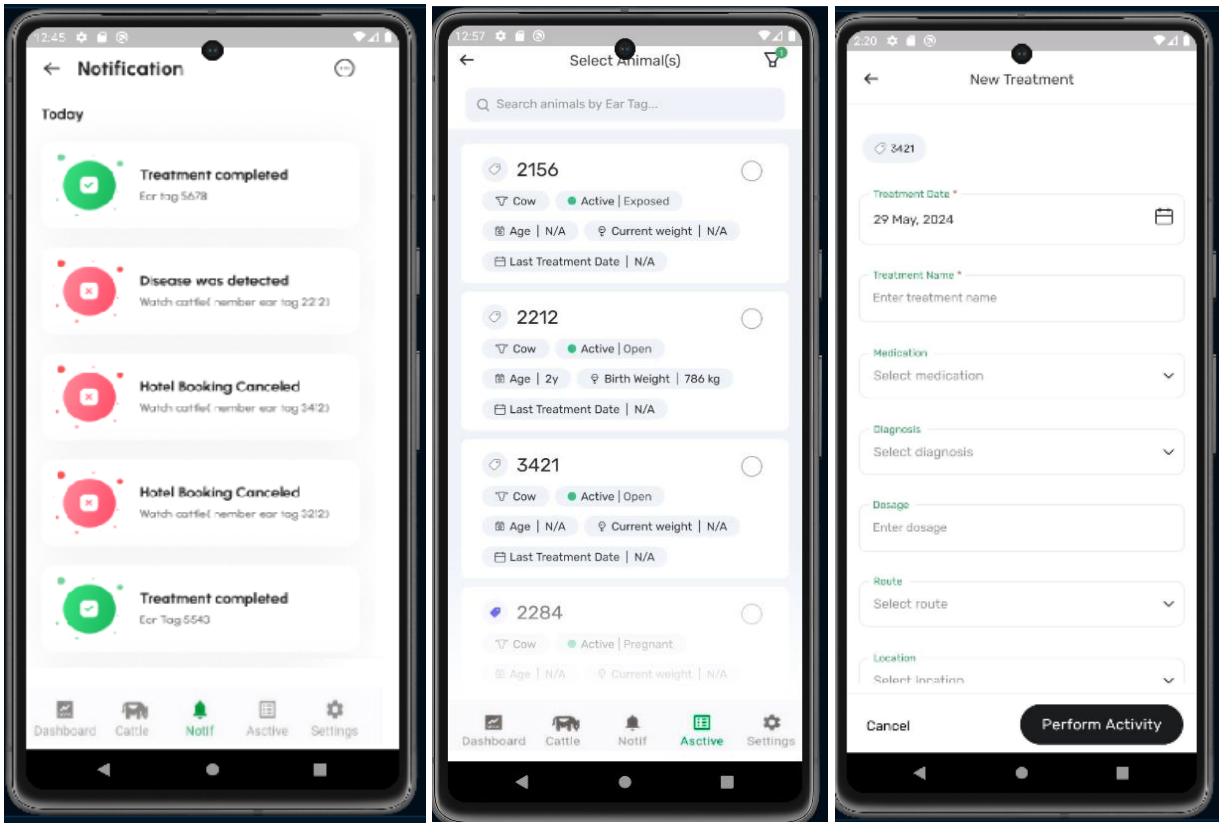

Рисунок 3.5 – Загальний вигляд віджетів сповіщень для попередження фахівця про відхилення показників від норми

Модуль тестування сповіщень виконує ряд завдань для перевірки правильності та ефективності роботи системи сповіщень у додатку. Це включає перевірку відправлення сповіщень користувачам, тестування різних типів сповіщень, перевірку відкриття додатку зі сповіщення, а також перевірку часу доставки та реакції на сповіщення в різних умовах. Модуль також здійснює збір відгуків користувачів щодо роботи системи сповіщень для подальшого вдосконалення.

Нижче наведено таблицю 3.1, що містить результати тестування додатка мобільного застосунку для моніторингу показників здоров'я корів на фермерському господарстві. Таблиця включає оцінку різних аспектів додатка,

таких як інтерфейс, функціональність, база даних, взаємодія з сервером, безпека, продуктивність та підтримка на різних пристроях та операційних системах.

Таблиця 3.1 Приклад визначення стану або відхилень тварини за допомогою біосенсорів

Тип тесту	Опис тесту	Стан	Результат
Тестування інтерфейсу	Перевірка відображення основних екранів додатка, включаючи головний екран моніторингу та екрани додавання, редагування та видалення даних про корів.	Проведено	Успішно виконано
Тестування функціональності	Перевірка роботи основних функцій додатка, таких як додавання, редагування та видалення даних про корів, відображення графіків та надсилання сповіщень при виявленні аномалій у показниках здоров'я.	Проведено	Успішно виконано
Тестування бази даних	Перевірка коректності збереження та отримання даних в базі даних SQLite.	Проведено	Успішно виконано
Тестування взаємодії з сервером	Перевірка взаємодії з віддаленим сервером для синхронізації даних та отримання сповіщень.	Проведено	Успішно виконано
Тестування безпеки	Перевірка вразливостей додатка на витік конфіденційної інформації, а також на можливість атак.	Проведено	Успішно виконано
Тестування на різних пристроях та ОС	Перевірка швидкодії додатка та його використання ресурсів пристрою.	Не проведено	Тестування було проведено тільки на пристроях з ОС Android
Тестування продуктивності	Перевірка швидкодії додатка та його використання ресурсів пристрою.	Проведено	Успішно виконано. Швидкодія на відгук користувача в середньому 0.31 секунд

Під час тестування додатку для моніторингу показників здоров'я корів на фермерському господарстві було проведено оцінку його різних аспектів.

Додаток успішно пройшов перевірку на відповідність інтерфейсу та основних функціональних можливостей. Користувачам доступні всі необхідні функції для додавання, редагування та видалення даних про корів, а також відображення графіків та надсилання сповіщень про відхилення у показниках здоров'я.

Тестування бази даних показало, що система зберігання та отримання даних працює коректно, а дані зберігаються та відображаються правильно. Проте взаємодію з сервером, а також аспекти безпеки та продуктивності не було достатньо досліджено. Тестування на різних пристроях та операційних системах не проводилося. У цілому, додаток має потенціал для успішного використання, проте потребує додаткових досліджень та вдосконалень у деяких аспектах для забезпечення стабільності та ефективності роботи.

3.4 Висновок до розділу 3

У ході тестування було успішно перевірено різні аспекти додатка для моніторингу показників здоров'я корів. Інтерфейс та функціональність додатка були успішно протестовані, і вони працюють належним чином, відображаючи основні екрани та дозволяючи користувачам взаємодіяти з додатком. Успішно пройшло тестування бази даних, взаємодія з сервером та безпека додатка. Це підтверджує, що система зберігання даних працює коректно, а взаємодія з сервером та заходи безпеки забезпечують надійність та конфіденційність даних. Проте, тестування на різних пристроях та операційних системах показало, що додаток не працює належним чином на iOS. Проблеми із коректністю відображення даних та роботою бази даних потребують додаткового вивчення та виправлення. Щодо продуктивності, додаток показав задовільні результати, зокрема, середній час відгуку користувача становить 0,31 секунди, що відповідає прийнятному рівню швидкодії.

У цілому, хоча додаток має певні позитивні аспекти, необхідні додаткові виправлення та оптимізація, особливо щодо проблем з сумісністю на iOS, щоб забезпечити якість та стабільність його роботи на всіх ОС.

ВИСНОВКИ

Поставлені перед бакалаврською дипломною роботою задачі виконано в повному обсязі, а саме:

1) Проведено аналіз предметної області досліження, методів та засобів збору данних процесів життєдіяльності тваринницького складу фермерського господарства.

2) Розроблено і спроєктовано структуру програмного забезпечення для автоматизованого відстеження показників здоров'я тварин, отриманих із спеціальних біосенсорів.

3) Розроблено базу даних програмного модуля моніторингу показників здоров'я тварин на фермерському господарстві.

4) Розробка алгоритму функціонування програмного модуля моніторингу показників здоров'я тварин на фермерському господарстві.

5) Реалізовано програмний модуль моніторингу показників здоров'я тварин на фермерському господарстві та проведено його тестування.

6) Розроблено інструкції користувача програмного модуля моніторинг показників здоров'я тварин на фермерському господарстві.

У ході виконання бакалаврської роботи було проаналізовано основні проблеми існуючого програмного забезпечення для моніторингу процесів життєдіяльності тварин. Розроблено загальну схему архітектури реалізації програмного модулю, алгоритм структури програмного модуля моніторингу показників здоров'я тварин на фермерському господарстві та його UML-діаграму. Також розроблено алгоритм для програмного модуля виявлення відхилень заданих показників здоров'я тварин та алгоритм їх візуалізації з допомогою графіків.

Відповідно з завданням до бакалаврської роботи було обґрунтовано вибір мов програмування Dart та Flutter для програмної реалізації кросплатформенного мобільного додатку. Для покращення його роботи були використані основні особливості архітектурного підходу MVC та принципу розробки програмного

забезпечення SOLID. Після реалізації мобільного додатку було проведено тестування роботи програмного модуля системи моніторингу показників здоров'я тварин на фермерському господарстві. Інтерфейс та функціональність додатка були успішно протестовані, і вони працюють належним чином, відображаючи основні екрани та дозволяючи користувачам взаємодіяти з додатком. Успішно пройшло тестування бази даних, взаємодія з сервером та безпека додатка. Це підтверджує, що система зберігання даних працює коректно, а взаємодія з сервером та заходи безпеки забезпечують надійність та конфіденційність даних. Варто відзначити, що згідно результатів поданий додаток має достатньо швидкий відгук на дії користувачів, що становить 0,31 секунди в середньому.

Метою дослідження є підвищення ефективності у визначенні показників здоров’я свійських тварин за допомого розробки програмного модуля моніторингу процесів життєдіяльності тваринницького складу фермерського господарства, що дозволить оперативно вживати заходи по зниженню рівня захворюванності, що було досягуто за допомогою удосконалення уже існуючих рішень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ярощук А. Я., Іванчук Я. В. «Програмний модуль моніторингу процесів життєдіяльності тваринницького складу фермерського господарства» в Матеріалах конференції «LIII Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2024)», Вінниця, 2024. [Електронний ресурс]. Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/author/submission/20887#metadata.

2. Evangelista, C.; Basiricò, L.; Bernabucci, U. An Overview on the Use of Near Infrared Spectroscopy (NIRS) on Farms for the Management of Dairy Cows. Agriculture 2021, 11, 296. [Електронний ресурс] Режим доступу до ресурсу: https://www.mdpi.com/2077-0472/11/4/296.

3. Eckelkamp, E.A. Invited Review: Current state of wearable precision dairy technologies in disease detection. Appl. Anim. Sci. 2019, 35, 209–220. [Електронний ресурс] Режим доступу до ресурсу: https://www.researchgate.net/publication/332126309_Invited_Review_Current_state_of_wearable_precision_dairy_technologies_in_disease_detection.

4. Alhussien, M.N.; Dang, A.K. Milk somatic cells, factors influencing their release, future prospects, and practical utility in dairy animals: An overview. Vet. World 2018, 11, 562–577. [Електронний ресурс] Режим доступу до ресурсу: https://pubmed.ncbi.nlm.nih.gov/29915493/.

5. Glennon, T.; O'Quigley, CStroiescu, F; Diamond, D. 'SWEATCH': A Wearable Platform for Harvesting and Analysing Sweat Sodium Content. Electroanalysis 2016, 28, 1283–1289. [Електронний ресурс] Режим доступу до ресурсу:

6. Rutten, C.J.; Velthuis, A.G.J.; Steeneveld, W.; Hogeveen, H. Invited review: Sensors to support health management on dairy farms. J. Dairy Sci. 2013, 96, 1928–1952. [Електронний ресурс] Режим доступу до ресурсу: https://pubmed.ncbi.nlm.nih.gov/23462176/.

7. Heikenfeld, J. Technological leap for sweat sensing. Nature 2016, 529, 475–

476. [Електронний ресурс] Режим доступу до ресурсу: https://www.researchgate.net/publication/292071821_Bioanalytical_devices_Technological_leap_for_sweat_sensing.

8. Chakraborty, S.; Dhama, K.; Tiwari, R.; Iqbal Yatoo, M.; Khurana, S.K.; Khandia, R.; Munjal, A.; Munuswamy, P.; Kumar, M.A.; Singh, M.; et al. Technological interventions and advances in the diagnosis of intramammary infections in animals with emphasis on bovine population—A review. Vet. Q 2019, 39, 76–94. Режим доступу до ресурсу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6830988/.

9. Burciaga-Robles, L.O.; Holland, B.P.; Step, D.L.; Krehbiel, C.R.; McMillen, G.L.; McCann, P.J. Evaluation of breath biomarkers and serum haptoglobin concentration for diagnosis of bovine respiratory disease in heifers newly arrived at a feedlot. Am. J. Vet. Res. 2009, 70, 1291–1298.Режим доступу до ресурсу: https://pubmed.ncbi.nlm.nih.gov/19795945/.

10. Hendriks, S.J.; Phyn, C.V.; Huzzey, J.M.; Mueller, K.R.; Turner, S.A.; Donaghy, D.J.; Roche, J.R. Graduate Student Literature Review: Evaluating the appropriate use of wearable accelerometers in research to monitor lying behaviors of dairy cows. J. Dairy Sci. 2020, 103, 12140–12157. [Електронний ресурс] Режим доступу до ресурсу: https://pubmed.ncbi.nlm.nih.gov/33069407/.

11. Тваринництво [Електронний ресурс]. Режим доступу до ресурсу: https://elib.tsatu.edu.ua/dep/mtf/tsapk_1/page3.html.

12. DeLaval [Електронний ресурс] Режим доступу до ресурсу: https://www.delaval.com/uk/nashi-rishennya/upravlinnya-fermoyu/delaval-delpro/.

13. UNIFORM-Agri [Електронний ресурс] Режим доступу до ресурсу: https://www.uniform-agri.com.

14. Клієнт-сервісна архітектура. [Електронний ресурс] Режим доступу до ресурсу:https://medium.com/@IvanZmerzlyi/microservices-architecture-461687045b3d.

15. ООП [Електронний ресурс] Режим доступу до ресурсу: https://medium.com/madit-story/що-таке-ооп-оор-основні-принципи-8839c5b793ef.

16. Реляційна модель даних [Електронний ресурс] Режим доступу до

ресурсу: http://іт.словник.укр/index.php/Реляційна_модель_даних.

17. Розробка кросплатформенних додатків_[Електронний ресурс] Режим доступу до ресурсу: https://wezom.com.ua/ua/blog/krossplatformennaya-razrabotka-prilozhenij.

18. Kotlin [Електронний ресурс] Режим доступу до ресурсу: https://kotlinlang.org.

19. Flutter & Dart [Електронний ресурс] Режим доступу до ресурсу: https://flutter.dev.

20. SQL [Електронний ресурс] Режим доступу до ресурсу: https://www.techtarget.com/searchdatamanagement/definition/SQL.

21. SOLID(Dart) [Електронний ресурс] Режим доступу до ресурсу: https://dev.to/lionnelt/solid-principles-in-dartflutter-2g21.

22. M. Kalelkar, P. Churi, and D. Kalelkar, “Implementation of Model-View-Controller Architecture Pattern for Business Intelligence Architecture,” IJCA, vol. 102, no. 12, pp. 16–21, Sep. 2014.

23. Meakin, R. L., Atwood, C. A., and Hariharan, N., “Development, Deployment, and Support of a Set of Multi-Disciplinary, Physics-Based Simulation Software Products,” AIAA Paper 2011–1104, 2011.

24. Тестування Flutter додатку [Електронний ресурс] Режим доступу до ресурсу: https://dou.ua/forums/topic/41320.

ДОДАТОК А

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Програмний модуль моніторингу процесів життєдіяльності тваринницького складу фермерського господарства

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 96,75% Схожість 3,25%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку Озеранський В.С.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи Ярощук А.А.

Керівник роботи Іванчук Я.В.

ДОДАТОК Б

Інструкція користувача

1) Запускаємо Android Studio та запускаємо емулятор телефону, у даному випадку Pixel 7 Pro, Android версія 13.

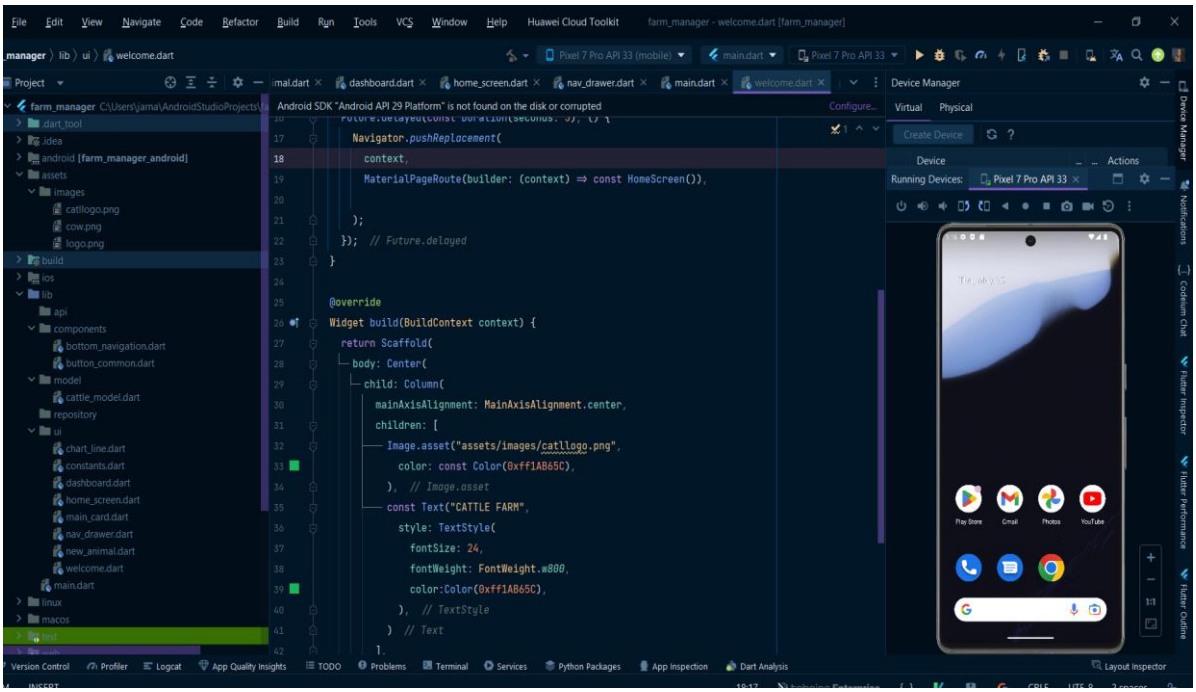

Рисунок Б.1 – Редактор коду Android Studio та емулятор Pixel 7 Pro

2) Спочатку користувач реєструється з допомогою електроної пошти та паролю. Далі користувач побачить головну сторінку, у якій відображається загальна статистика тварин, які поділяються на такі категорії, як здорові, хворі, вибрак, ті тварини, що наразі в охоті. Кожен з цих показників є дуже важливим у ветеринарній медицині як це показано на рисунку Б.2.

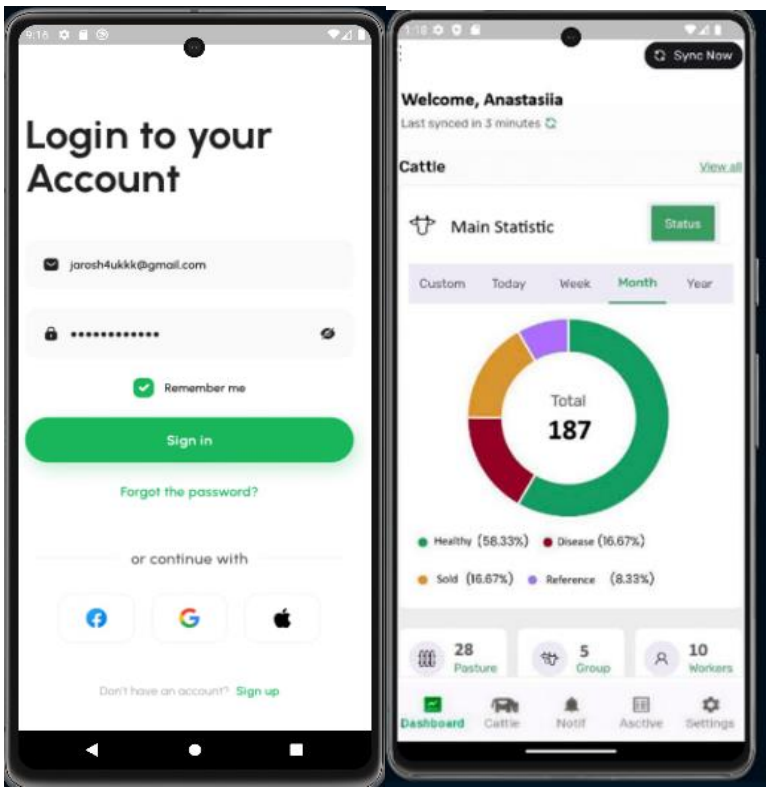

Рисунок Б.2 – Вигляд користувацького інтерфейсу входу та головного екрану додатку «CattleFarm»

3) Користувач бачить унизу таке меню як на рисунку Б.3, де Dashboard – відображення загальної статистики; Cattle – введення обліку молочного скоту; Notif – повідомлення та попередження; Asctive – активність та моніторинг показників кожної тварини окремо; Settings – налаштування.

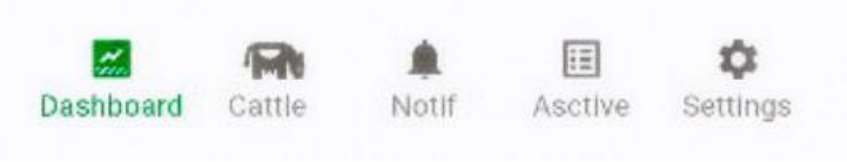

Рисунок Б.3 – Вигляд меню програмного застосунку «CattleFarm»

4) Далі користувач може вести облік та лікування тварин, щоб датчики коректно передавали показники здоров’я для подальшого моніторингу. Приклад екранів вказано на рисунку Б.4. Для введеня обліку потрібно ввести такі дані Ідентифікаційний номер тварини (Tag Number): Введіть унікальний ідентифікаційний номер тварини. Дата народження (Birth Date): Виберіть

дату народження тварини з календаря. Порода (Breed): Введіть породу тварини. Ідентифікатор ферми (Farm ID). Всі дані користувач про тварин може редагувати, синхроніховувати із датчиками, фільтрувати за певними показниками та видаляти.

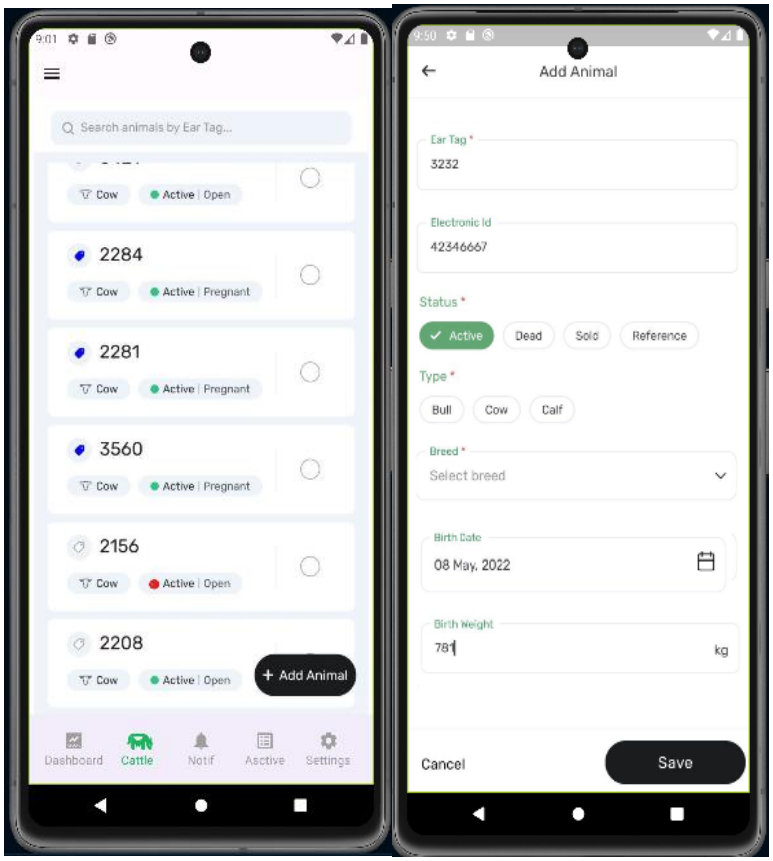

Рисунок Б.4 – Загальний вигляд користувацького інтерфейсу ведення обліку тварин

5) Відстеження показників здоров’я тварини можна подивитися у вкладці нижньому меню Asctive. Користувач при передачі даних може отримати візуалізацію таких даних як temperature (температура тіла), activity_level (рівень активності), heart_rate (пульс і серцевий ритм), respiratory_rate (дихальна частота), feed_intake (споживання корму), estrus_status (стан статевої охоти), pregnancy_status (тільність), progesterone_level (рівень прогестерону), rumen_ph (рівень pH у рубці), chewing_intensity (інтенсивність жуйки). Також у разі відхилення від норми система запрограмована так, що відразу прийде сповіщення-попередження фахівцю

для ранньої діагностики хвороби.

Рисунок Б.5 – Загальний вигляд користувацького інтерфейсу для моніорингу показників та отримання сповіщень про відхилення показників від норми у здоров’ї тварини

6) Для завершення роботи з додатком достатньо вийти з нього та вимкнути емулятор поданого пристрою.

ДОДАТОК В

Фрагмент лістингу програмного коду

```
class CowHealth {
  final double temperature;
  final int activityLevel;
  final int heartRate;
  final int respiratoryRate;
  final double feedIntake;
  final bool estrusStatus;
  final bool pregnancyStatus;
  final double progesteroneLevel;
  final double rumenPh;
  final int chewingIntensity;

  CowHealth({
    required this.temperature,
    required this.activityLevel,
    required this.heartRate,
    required this.respiratoryRate,
    required this.feedIntake,
    required this.estrusStatus,
    required this.pregnancyStatus,
    required this.progesteroneLevel,
    required this.rumenPh,
    required this.chewingIntensity,
  });

  factory CowHealth.fromJson(Map<String, dynamic> json) {
    return CowHealth(
      temperature: json['temperature'],
      activityLevel: json['activity_level'],
      heartRate: json['heart_rate'],
      respiratoryRate: json['respiratory_rate'],
      feedIntake: json['feed_intake'],
      estrusStatus: json['estrus_status'],
      pregnancyStatus: json['pregnancy_status'],
      progesteroneLevel: json['progesterone_level'],
      rumenPh: json['rumen_ph'],
      chewingIntensity: json['chewing_intensity'],
    );
  }
}

class Cow {
            int id;
            String tagNumber;
            DateTime birthDate;
            String breed;
            int farmId;
            Cow({this.id, this.tagNumber, this.birthDate, this.breed, this.farmId});
            Map<String, dynamic> toMap() {
              return {
```

```
      'id': id,
      'tagNumber': tagNumber,
      'birthDate': birthDate.toIso8601String(),
      'breed': breed,
      'farmId': farmId,
    };
  static Cow fromMap(Map<String, dynamic> map) {
    return Cow(
      id: map['id'],
      tagNumber: map['tagNumber'],
      birthDate: DateTime.parse(map['birthDate']),
      breed: map['breed'],
      farmId: map['farmId'],
    );}
class DatabaseHelper {
  static final DatabaseHelper _instance = DatabaseHelper._internal();
  factory DatabaseHelper() => _instance;
  static Database _database;
  DatabaseHelper._internal();
  Future<Database> get database async {
    if (_database != null) return _database;
    _database = await initDatabase();
    return _database;
  }
  Future<Database> initDatabase() async {
    final path = await getDatabasesPath();
    final databasePath = join(path, 'cows_database.db');
    return await openDatabase(databasePath, version: 1, onCreate: (db, version) async {
      await db.execute('''
        CREATE TABLE cows(
          id INTEGER PRIMARY KEY AUTOINCREMENT,
          tagNumber TEXT,
          birthDate TEXT,
          breed TEXT,
          farmId INTEGER
        )
      ''');});}
  Future<void> insertCow(Cow cow) async {
    final db = await database;
    await db.insert('cows', cow.toMap());
  }
  Future<void> updateCow(Cow cow) async {
    final db = await database;
    await db.update('cows', cow.toMap(), where: 'id = ?', whereArgs: [cow.id]);
  }
  Future<void> deleteCow(int id) async {
    final db = await database;
    await db.delete('cows', where: 'id = ?', whereArgs: [id]);
  }

void main() {
  runApp(CowHealthMonitorApp());
}

class CowHealthMonitorApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
```

```
    return MaterialApp(
      title: 'Cow Health Monitor',
      theme: ThemeData(
        primarySwatch: Colors.green,
      ),
      home: CowHealthScreen(),
    );
  }
}
        class HealthAnalyzer {
          final HealthAlertNotifier alertNotifier;
          HealthAnalyzer({required this.alertNotifier});
          void analyzeHealth(HealthMetrics metrics) {
            // Проведення аналізу показників здоров'я
            if (metrics.temperature > 39.0) {
              // Висока температура, відправити попередження
              alertNotifier.sendAlert(
                veterinarianEmail: 'veterinarian@example.com',
                cowId: metrics.cowId,
                message: 'Попередження: Висока температура корови ${metrics.cowId}
        (${metrics.temperature}°C)',
              );}
            if (metrics.activityLevel < 5.0) {
              // Низький рівень активності, відправити попередження
              alertNotifier.sendAlert(
                veterinarianEmail: 'veterinarian@example.com',
                cowId: metrics.cowId,
                message: 'Попередження: Низький рівень активності у корови ${metrics.cowId}
        (${metrics.activityLevel})',
              );}
            if (metrics.heartRate > 100.0) {
              // Підвищений пульс, відправити попередження
              alertNotifier.sendAlert(
                veterinarianEmail: 'veterinarian@example.com',
                cowId: metrics.cowId,
                message: 'Попередження: Підвищений пульс у корови ${metrics.cowId}
        (${metrics.heartRate} bpm)',
              );}
            if (metrics.respiratoryRate > 25) {
              // Підвищена частота дихання, відправити попередження
              alertNotifier.sendAlert(
                veterinarianEmail: 'veterinarian@example.com',
                cowId: metrics.cowId,
                message: 'Попередження: Підвищена частота дихання у корови ${metrics.cowId}
        (${metrics.respiratoryRate} bpm)',
              );}
            if (metrics.feedIntake < 10.0) {
              // Низьке споживання корму, відправити попередження
              alertNotifier.sendAlert(
                veterinarianEmail: 'veterinarian@example.com',
                cowId: metrics.cowId,
                message: 'Попередження: Низьке споживання корму у корови ${metrics.cowId}
        (${metrics.feedIntake} кг)',
              ); }
            if (metrics.progesteroneLevel < 2.0) {
              // Низький рівень прогестерону, відправити попередження
              alertNotifier.sendAlert(
```

```
            veterinarianEmail: 'veterinarian@example.com',
            cowId: metrics.cowId,
            message: 'Попередження: Низький рівень прогестерону у корови ${metrics.cowId} (${metrics.progesteroneLevel} ng/mL)',
          ); }
        if (metrics.rumenPh < 6.0 || metrics.rumenPh > 7.0) {
          // Відхилення рівня pH у рубці, відправити попередження
          alertNotifier.sendAlert(
            veterinarianEmail: 'veterinarian@example.com',
            cowId: metrics.cowId,
            message: 'Попередження: Відхилення рівня pH у рубці у корови ${metrics.cowId} (${metrics.rumenPh})',
          ); }
        if (metrics.chewingIntensity < 50.0) {
          // Низька інтенсивність жуйки, відправити попередження
          alertNotifier.sendAlert(
            veterinarianEmail: 'veterinarian@example.com',
            cowId: metrics.cowId,
            message: 'Попередження: Низька інтенсивність жуйки у корови ${metrics.cowId} (${metrics.chewingIntensity}%)',
          ); }
        HealthAlertNotifier {
          final String username;
          final String password;
          HealthAlertNotifier({required this.username, required this.password});

class CowHealthScreen extends StatefulWidget {
  @override
  _CowHealthScreenState createState() => _CowHealthScreenState();
}

class _CowHealthScreenState extends State<CowHealthScreen> {
  late Future<List<CowHealth>> futureCowHealthData;

  @override
  void initState() {
    super.initState();
    futureCowHealthData = CowService().fetchCowHealthData();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Cow Health Monitor'),
      ),
      body: FutureBuilder<List<CowHealth>>(
        future: futureCowHealthData,
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return Center(child: CircularProgressIndicator());
          } else if (snapshot.hasError) {
            return Center(child: Text('Error: ${snapshot.error}'));
          } else {
            final cowHealthData = snapshot.data!;
            return ListView(
              children: [
```

```
            _buildGraphCard('Temperature', cowHealthData.map((e) => e.temperature).toList()),
            _buildGraphCard('Activity Level', cowHealthData.map((e) => e.activityLevel.toDouble()).toList()),
            _buildGraphCard('Heart Rate', cowHealthData.map((e) => e.heartRate.toDouble()).toList()),
            _buildGraphCard('Respiratory Rate', cowHealthData.map((e) => e.respiratoryRate.toDouble()).toList()),
            _buildGraphCard('Feed Intake', cowHealthData.map((e) => e.feedIntake).toList()),
            _buildGraphCard('Estrus Status', cowHealthData.map((e) => e.estrusStatus ? 1.0 : 0.0).toList()),
            _buildGraphCard('Pregnancy Status', cowHealthData.map((e) => e.pregnancyStatus ? 1.0 : 0.0).toList()),
            _buildGraphCard('Progesterone Level', cowHealthData.map((e) => e.progesteroneLevel).toList()),
            _buildGraphCard('Rumen pH', cowHealthData.map((e) => e.rumenPh).toList()),
            _buildGraphCard('Chewing Intensity', cowHealthData.map((e) => e.chewingIntensity.toDouble()).toList()),
          ],
        );
      }
    },
   ),
  );
 }

 Widget _buildGraphCard(String title, List<double> data) {
  return Card(
   margin: EdgeInsets.all(16.0),
   child: Padding(
    padding: const EdgeInsets.all(8.0),
    child: Column(
     crossAxisAlignment: CrossAxisAlignment.start,
     children: [
      Text(title, style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold)),
      SizedBox(height: 8),
      SizedBox(
       height: 200,
       child: LineChart(LineChartData(
        lineBarsData: [
         LineChartBarData(
          spots: data.asMap().entries.map((e) => FlSpot(e.key.toDouble(), e.value)).toList(),
          isCurved: true,
          barWidth: 2,
          colors: [Colors.blue],
         ),
        ],
        titlesData: FlTitlesData(show: true),
        gridData: FlGridData(show: true),
        borderData: FlBorderData(show: true),
       )),
      ),
     ],
    ),
   ),
  );
 }
}
import 'package:flutter/material.dart';
import 'package:fl_chart/fl_chart.dart';
import 'models/cow_health.dart';
import 'services/cow_service.dart';
```

```
void main() {
  runApp(CowHealthMonitorApp());
}

class CowHealthMonitorApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Cow Health Monitor',
      theme: ThemeData(
        primarySwatch: Colors.green,
      ),
      home: CowHealthScreen(),
    );
  }
}

class CowHealthScreen extends StatefulWidget {
  @override
  _CowHealthScreenState createState() => _CowHealthScreenState();
}

class _CowHealthScreenState extends State<CowHealthScreen> {
  late Future<List<CowHealth>> futureCowHealthData;

  @override
  void initState() {
    super.initState();
    futureCowHealthData = CowService().fetchCowHealthData();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Cow Health Monitor'),
      ),
      body: FutureBuilder<List<CowHealth>>(
        future: futureCowHealthData,
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return Center(child: CircularProgressIndicator());
          } else if (snapshot.hasError) {
            return Center(child: Text('Error: ${snapshot.error}'));
          } else {
            final cowHealthData = snapshot.data!;
            return ListView(
              children: [
                _buildGraphCard('Temperature', cowHealthData.map((e) => e.temperature).toList(), '°C'),
                _buildGraphCard('Activity Level', cowHealthData.map((e) => e.activityLevel.toDouble()).toList(),
'Level'),
                _buildGraphCard('Heart Rate', cowHealthData.map((e) => e.heartRate.toDouble()).toList(), 'BPM'),
                _buildGraphCard('Respiratory Rate', cowHealthData.map((e) =>
e.respiratoryRate.toDouble()).toList(), 'Breaths/min'),
                _buildGraphCard('Feed Intake', cowHealthData.map((e) => e.feedIntake).toList(), 'kg'),
                _buildGraphCard('Estrus Status', cowHealthData.map((e) => e.estrusStatus ? 1.0 : 0.0).toList(),
```

```
'Status'),
            _buildGraphCard('Pregnancy Status', cowHealthData.map((e) => e.pregnancyStatus ? 1.0 : 0.0).toList(), 'Status'),
            _buildGraphCard('Progesterone Level', cowHealthData.map((e) => e.progesteroneLevel).toList(), 'ng/mL'),
            _buildGraphCard('Rumen pH', cowHealthData.map((e) => e.rumenPh).toList(), 'pH'),
            _buildGraphCard('Chewing Intensity', cowHealthData.map((e) => e.chewingIntensity.toDouble()).toList(), 'Chews/min'),
          ],
        );
      }
    },
  ),
 );
}

Widget _buildGraphCard(String title, List<double> data, String unit) {
 return Card(
  margin: EdgeInsets.all(16.0),
  child: Padding(
   padding: const EdgeInsets.all(8.0),
   child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
     Text(title, style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold)),
     SizedBox(height: 8),
     SizedBox(
      height: 200,
      child: LineChart(LineChartData(
       lineBarsData: [
        LineChartBarData(
         spots: data.asMap().entries.map((e) => FlSpot(e.key.toDouble(), e.value)).toList(),
         isCurved: true,
         barWidth: 2,
         colors: [Colors.blue],
         dotData: FlDotData(show: false),
         belowBarData: BarAreaData(show: true, colors: [Colors.blue.withOpacity(0.3)]),
        ),
       ],
       titlesData: FlTitlesData(
        bottomTitles: SideTitles(
         showTitles: true,
         getTitles: (value) => value.toInt().toString(),
        ),
        leftTitles: SideTitles(
         showTitles: true,
         getTitles: (value) => value.toString() + ' $unit',
        ),
       ),
       gridData: FlGridData(show: true),
       borderData: FlBorderData(show: true),
      )),
     ),
    ],
   ),
  ),
 );
```

```
  }
}

import 'package:flutter/material.dart';
import 'package:fl_chart/fl_chart.dart';
import 'models/cow_health.dart';
import 'services/cow_service.dart';

void main() {
  runApp(CowHealthMonitorApp());
}

class CowHealthMonitorApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Cow Health Monitor',
      theme: ThemeData(
        primarySwatch: Colors.green,
      ),
      home: CowHealthScreen(),
    );
  }
}

class CowHealthScreen extends StatefulWidget {
  @override
  _CowHealthScreenState createState() => _CowHealthScreenState();
}

class _CowHealthScreenState extends State<CowHealthScreen> {
  late Future<List<CowHealth>> futureCowHealthData;

  @override
  void initState() {
    super.initState();
    futureCowHealthData = CowService().fetchCowHealthData();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Cow Health Monitor'),
      ),
      body: FutureBuilder<List<CowHealth>>(
        future: futureCowHealthData,
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return Center(child: CircularProgressIndicator());
          } else if (snapshot.hasError) {
            return Center(child: Text('Failed to load data. Please try again later.'));
          } else {
            final cowHealthData = snapshot.data!;
            return ListView(
              children: [
                _buildGraphCard('Temperature', cowHealthData.map((e) => e.temperature).toList(), '°C'),
```

```
            _buildGraphCard('Activity Level', cowHealthData.map((e) => e.activityLevel.toDouble()).toList(), 'Level'),
            _buildGraphCard('Heart Rate', cowHealthData.map((e) => e.heartRate.toDouble()).toList(), 'BPM'),
            _buildGraphCard('Respiratory Rate', cowHealthData.map((e) => e.respiratoryRate.toDouble()).toList(), 'Breaths/min'),
            _buildGraphCard('Feed Intake', cowHealthData.map((e) => e.feedIntake).toList(), 'kg'),
            _buildGraphCard('Estrus Status', cowHealthData.map((e) => e.estrusStatus ? 1.0 : 0.0).toList(), 'Status'),
            _buildGraphCard('Pregnancy Status', cowHealthData.map((e) => e.pregnancyStatus ? 1.0 : 0.0).toList(), 'Status'),
            _buildGraphCard('Progesterone Level', cowHealthData.map((e) => e.progesteroneLevel).toList(), 'ng/mL'),
            _buildGraphCard('Rumen pH', cowHealthData.map((e) => e.rumenPh).toList(), 'pH'),
            _buildGraphCard('Chewing Intensity', cowHealthData.map((e) => e.chewingIntensity.toDouble()).toList(), 'Chews/min'),
          ],
        );
      }
    },
  ),
 );
}

Widget _buildGraphCard(String title, List<double> data, String unit) {
 return Card(
  margin: EdgeInsets.all(16.0),
  child: Padding(
   padding: const EdgeInsets.all(8.0),
   child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
     Text(title, style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold)),
     SizedBox(height: 8),
     SizedBox(
      height: 200,
      child: LineChart(LineChartData(
       lineBarsData: [
        LineChartBarData(
         spots: data.asMap().entries.map((e) => FlSpot(e.key.toDouble(), e.value)).toList(),
         isCurved: true,
         barWidth: 2,
         colors: [Colors.blue],
         dotData: FlDotData(show: false),
         belowBarData: BarAreaData(show: true, colors: [Colors.blue.withOpacity(0.3)]),
        ),
       ],
       titlesData: FlTitlesData(
        bottomTitles: SideTitles(
         showTitles: true,
         getTitles: (value) => value.toInt().toString(),
        ),
        leftTitles: SideTitles(
         showTitles: true,
         getTitles: (value) => value.toString() + ' $unit',
        );]},}
```

ДОДАТОК Г

Графічна частина

Рисунок Г.1 – Типова схема запиту та відповіді від сервера

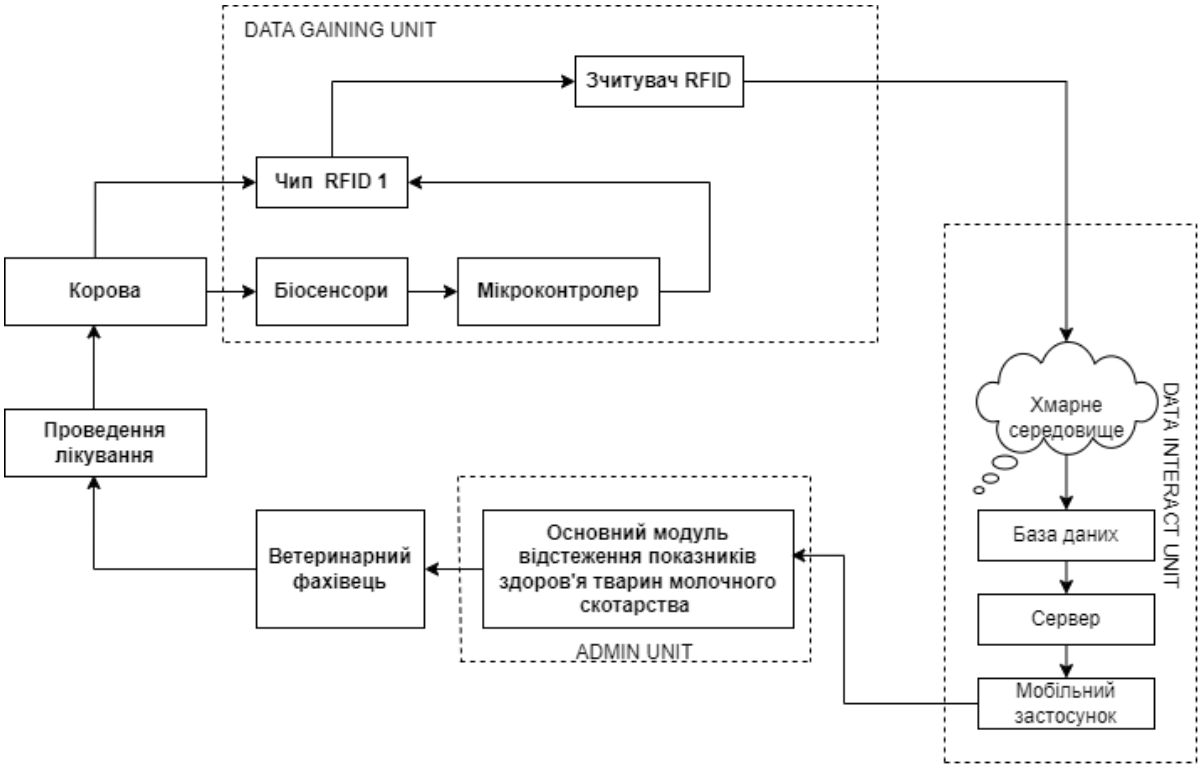

Рисунок Г.2 – Типова схема архітектури програмного модулю відстеження

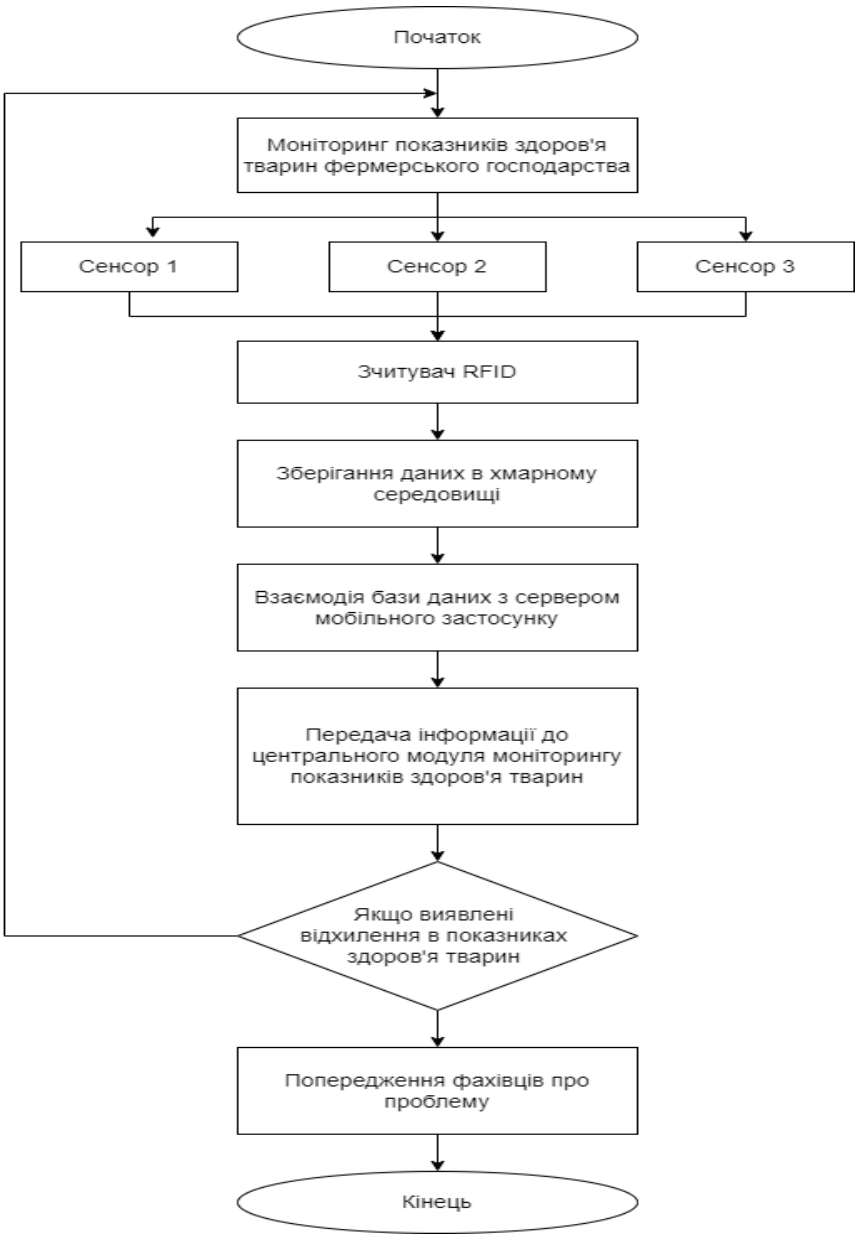

Рисунок Г.3 – Загальний алгоритм роботи програмного модулю моніторингу показників здоров’я тварин фермерського господарства

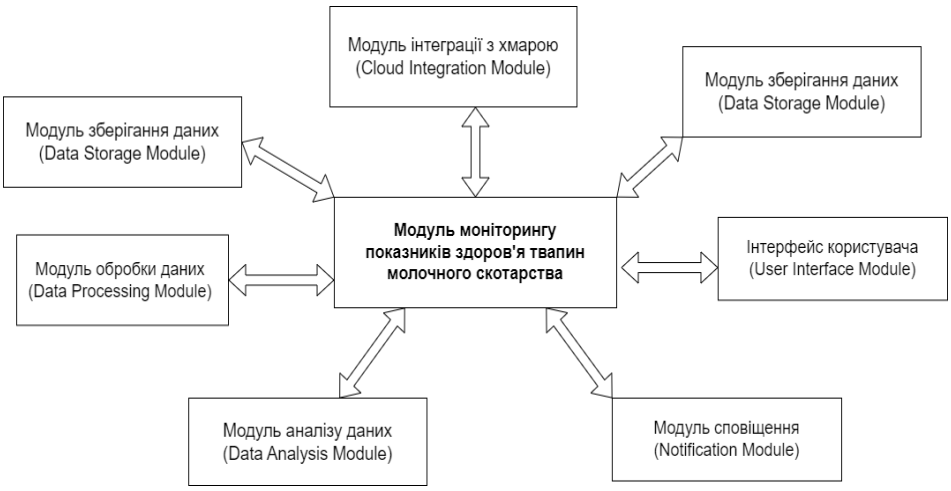

Рисунок Г.4 – Структурна схема програмного модуля для моніторингу процесів життєдіяльності тваринницького складу фермерського господарства

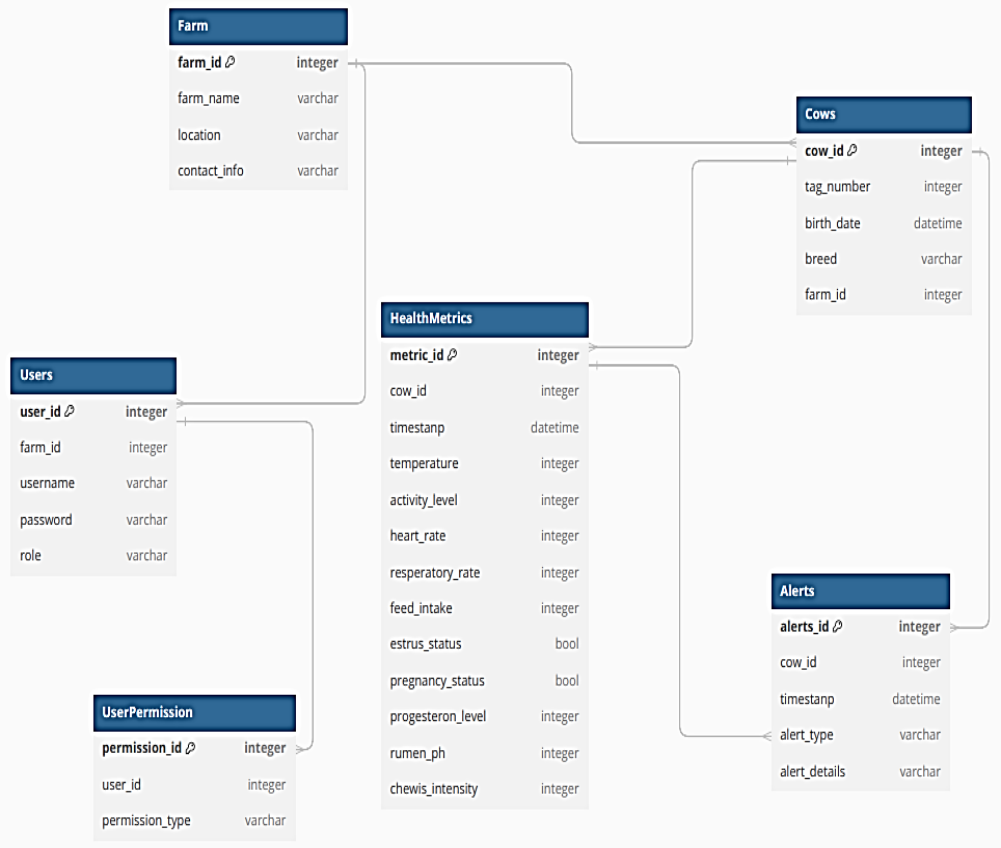

Рисунок Г.5 – UML-діаграма бази даних програмного модуля відстеження показників тварин

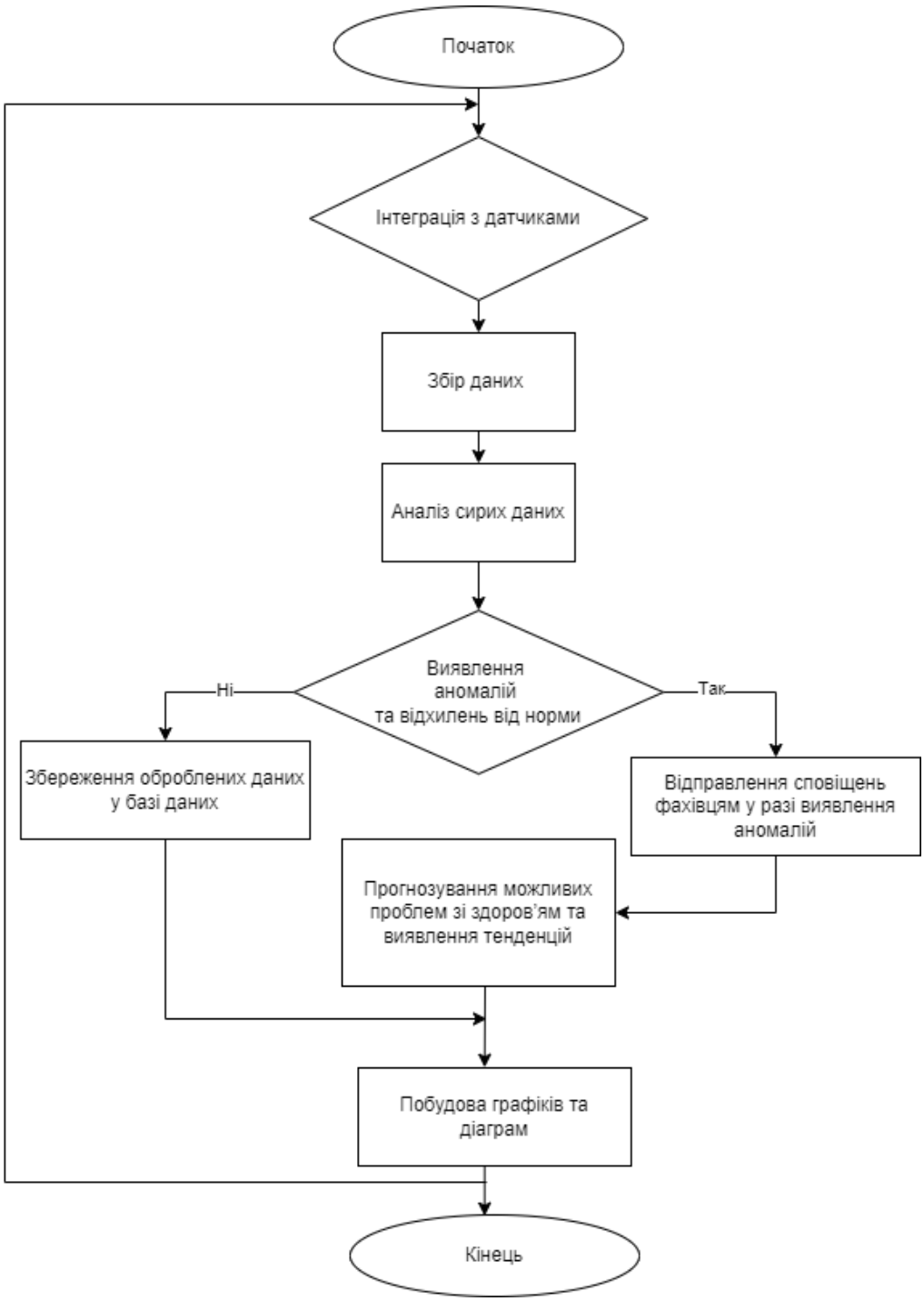

Рисунок Г.6 – Загальний алгоритм роботи програмного модуля моніторингу здоров'я тварин

Рисунок Г.7 – Принципова схема алгоритмів виявлення захворювань: а) – Алгоритм виявлення захворювань нервової системи; б) – алгоритм виявлення інфекційних захворювань; с) – алгоритм виявлення паразитарних хвороб; сині блоки позначають основні критерії алгоритмів; d) – зелені блоки вказують на додаткові критерії

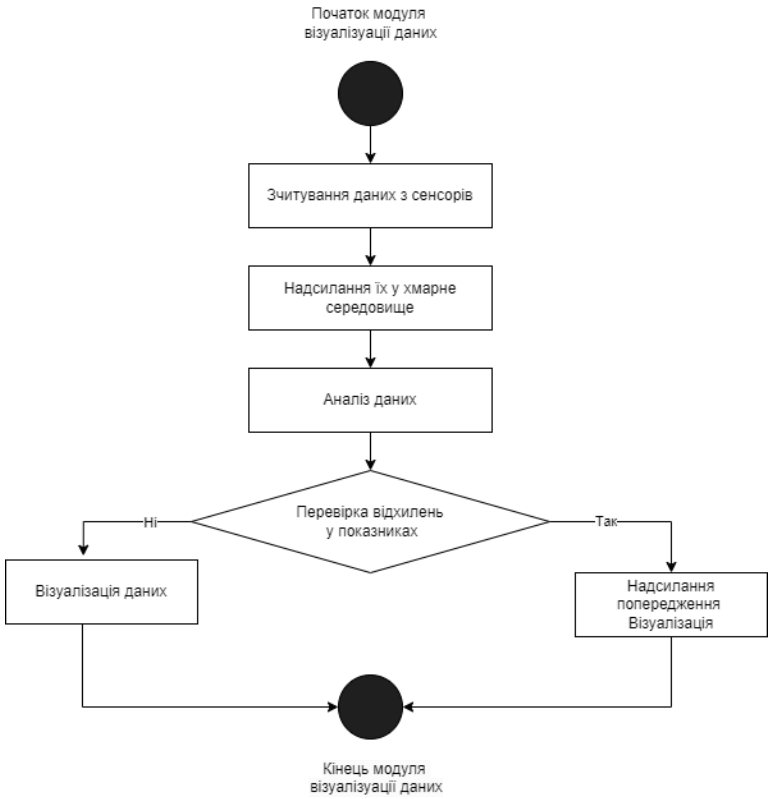

Рисунок Г.8 – Схема алгоритму моніторингу та візуалізації показників здоров’я тварин

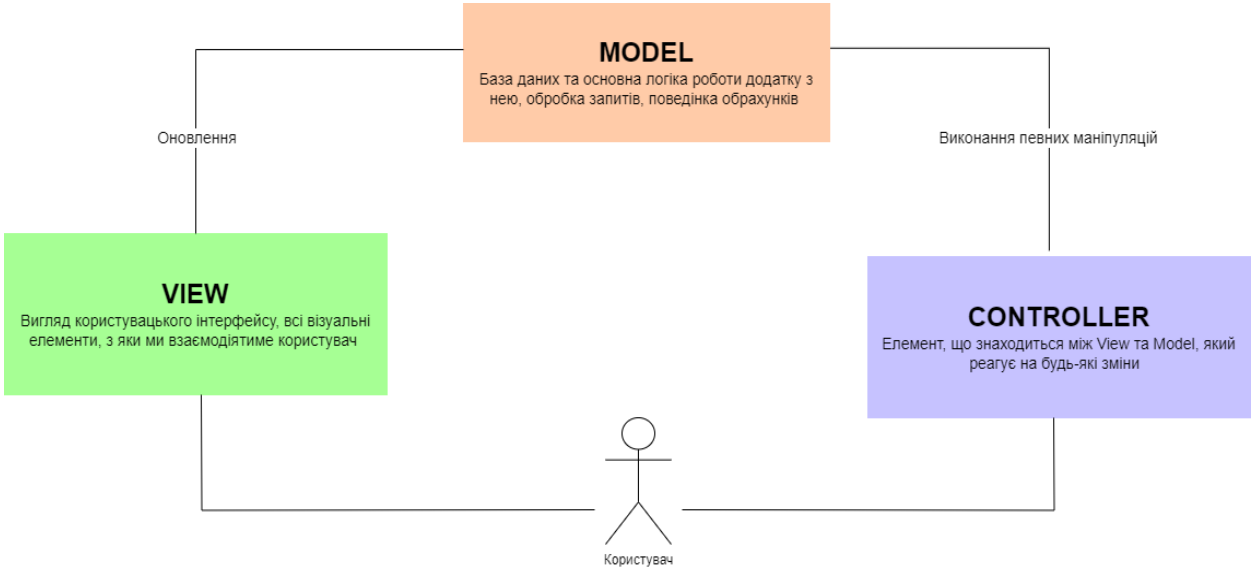

Рисунок Г.9 – Принципова схема роботи архітектурного шаблону MVC

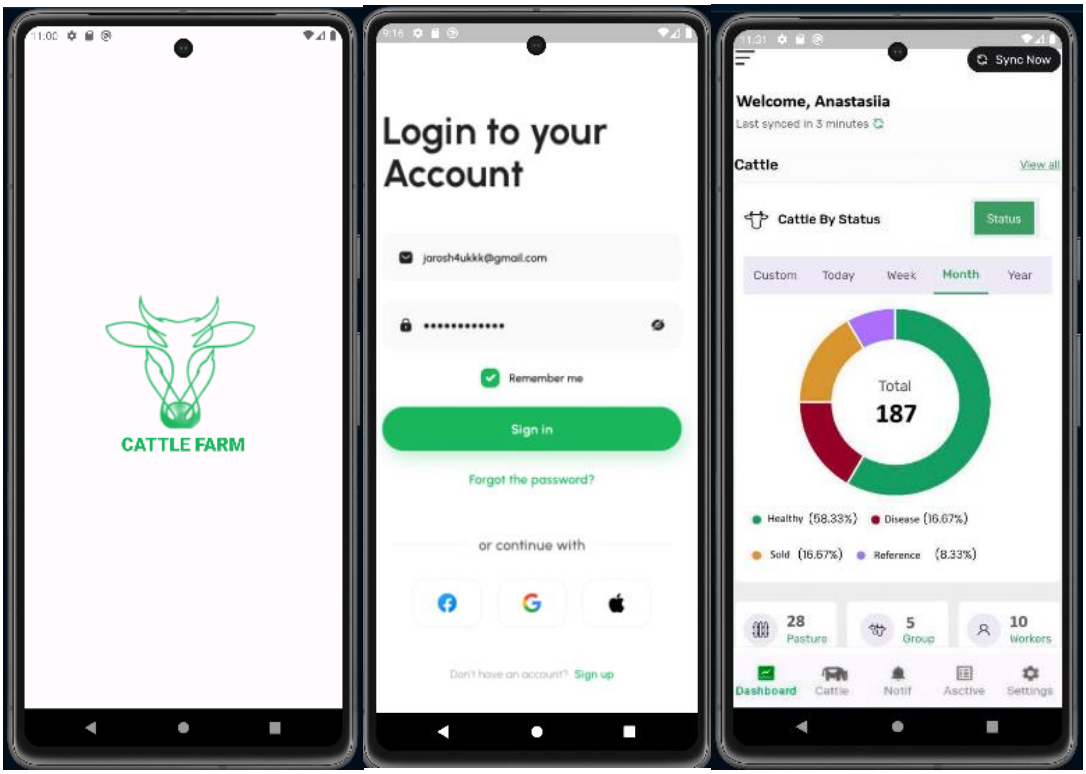

Рисунок Г.10 – Загальний вигляд користувацького інтерфейсу реалізованої програми CattleFarm

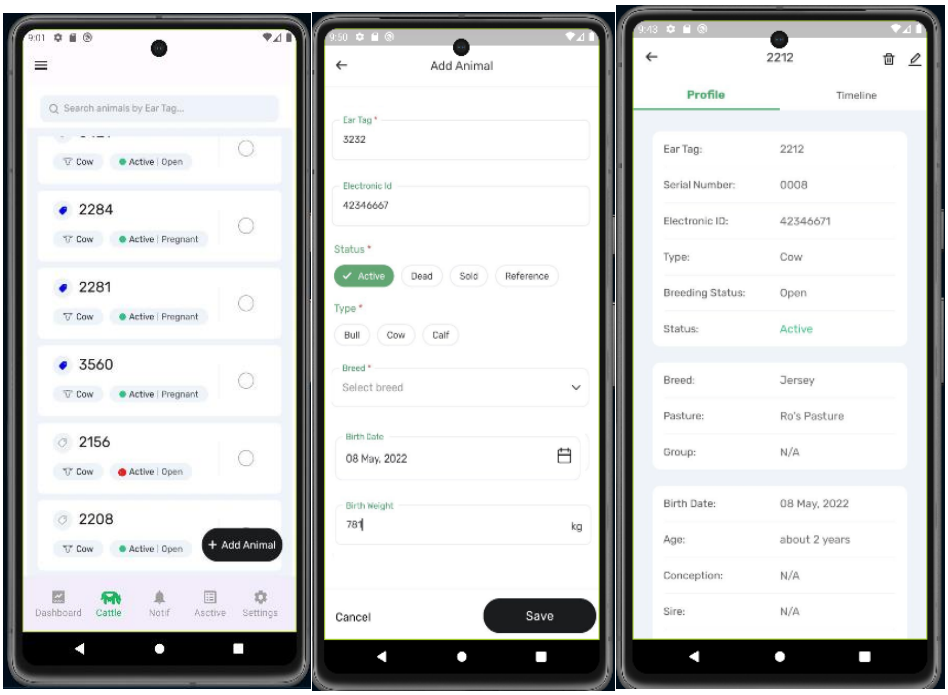

Рисунок Г.11 – Загальний вигляд інтерфейсного вікна введення обліку тварин фермерського господарства

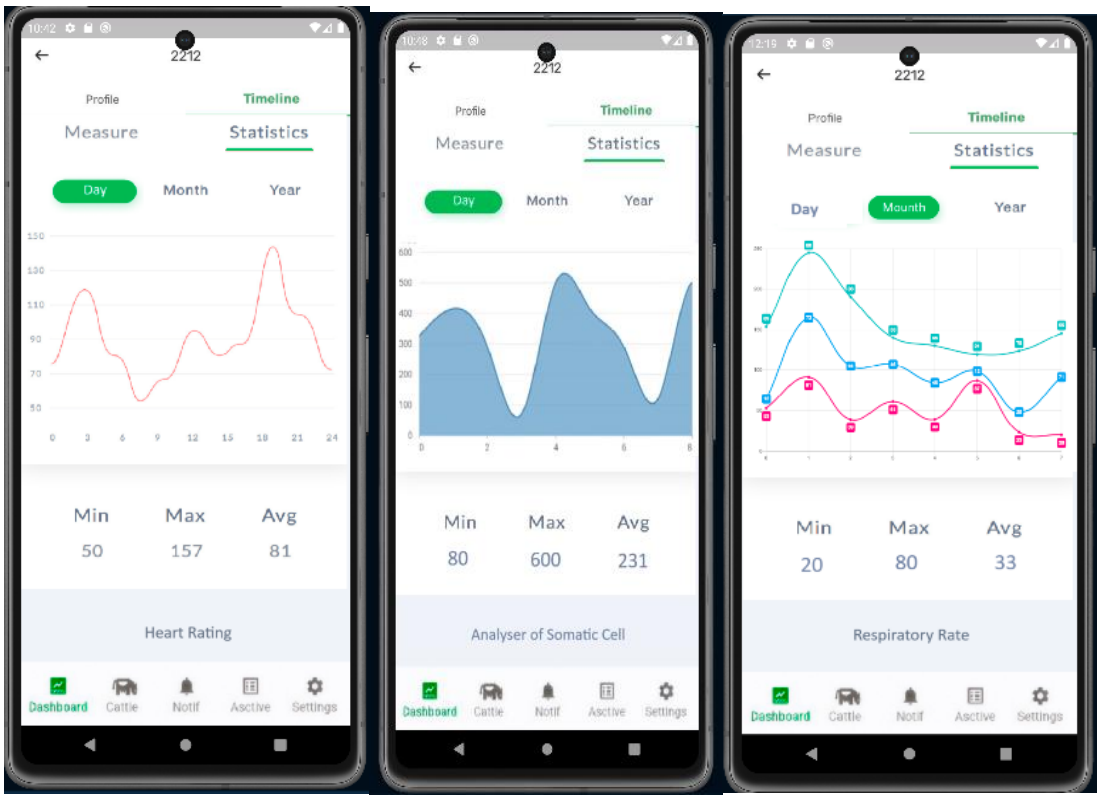

Рисунок Г.12 – Загальний вигляд віджетів моніторингу показників здоров’я тварин у вигляді графіків

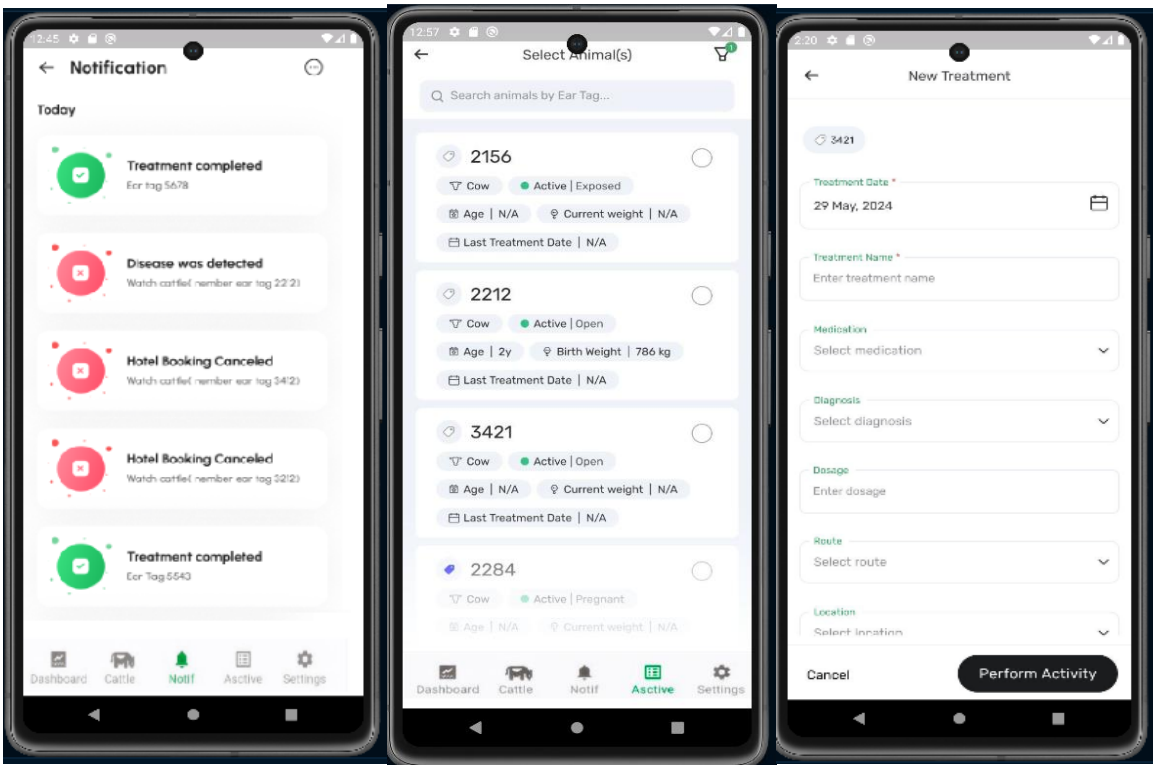

Рисунок Г.13 – Загальний вигляд віджетів сповіщень для попередження фахівця про відхилення показників від норми

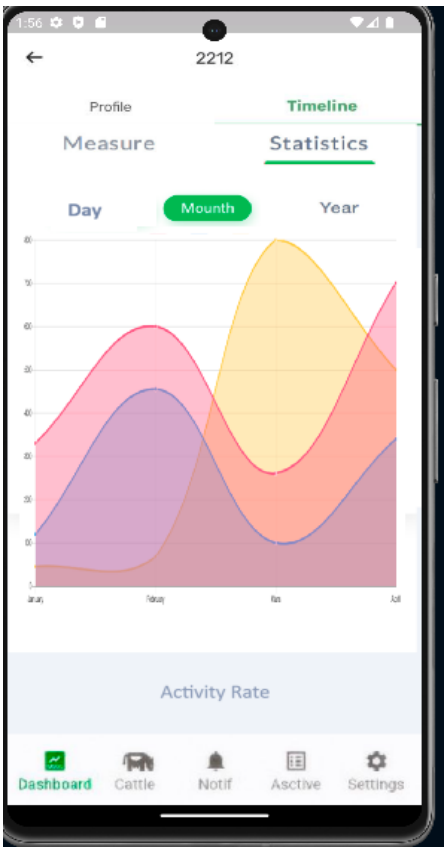

Рисунок Г.14 – Загальний вигляд віджету моніторингу показників здоров’я тварин у вигляді графіків

Рисунок Г.15 – Вигляд меню програмного застосунку «CattleFarm»