Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
магістр
(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ ВІДСЛІДКОВУВАННЯ СТАНУ ХУДОБИ. RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR CATTLE HEALTH MONITORING
(назва теми)

Виконав(ла): студент(ка) 2 курсу, групи КНТ-214м

Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Системи штучного інтелекту

ДУБИНА О.Ю.
(ПРІЗВИЩЕ та ініціали)

Керівник ЛЬОВКІН В.М.
(ПРІЗВИЩЕ та ініціали)

Рецензент ЗЕЛІК О.В.
(ПРІЗВИЩЕ та ініціали)

2025

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
«_____» _____2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ДУБИНИ Олега Юрійовича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів відслідковування стану худоби. Research and Software Implementation of Methods for Cattle Health Monitoring

керівник проєкту (роботи) к.т.н., доцент ЛЬОВКІН Валерій Миколайович,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від « 30 » вересня 2025 року № 447
2. Строк подання студентом проєкту (роботи) 08 грудня 2025 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Проєктування програмного забезпечення. 4. Основні рішення щодо реалізації програмного забезпечення. 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	ЛЬОВКІН В.М., доцент		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання «__30__» ____вересня____2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Дослідження проблеми	3-5 тижні	Розділ 2
4	Проєктування та розробка архітектури програми	6 тиждень	Розділ 3
5	Розробка програми	7-8 тижні	Розділ 4
6	Тестування та експериментальне дослідження	9-10 тижні	Розділ 5
7	Оформлення пояснювальної записки та документів до неї	11 тиждень	Додатки
8	Нормоконтроль та рецензування	12 тиждень	
9	Захист роботи	12 тиждень	

Студент(ка) _____ Олег ДУБИНА
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Валерій ЛЬОВКІН
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра: 114 с., 26 рис., 3 табл., 3 дод., 25 джерел.

ХУДОБА, ГОСПОДАРСТВО, УДІЙ, ПРОГНОЗУВАННЯ, ДЕРЕВО РІШЕНЬ, ВЕБЗАСТОСУНОК.

Об’єкт роботи – процес обчислення стану худоби.

Предмет роботи – комп'ютерні алгоритми та програмні засоби відслідковування стану худоби.

Мета роботи – збільшення точності прогнозування стану худоби шляхом розробки програмних засобів відслідковування стану худоби.

Матеріали, методи та технічні засоби: мова Python, фреймворк Django, засоби машинного навчання, зокрема дерева рішень, система MySQL.

Результати. Створена програма відслідковування стану худоби необхідна користувачам для керування даними худоби за тим господарством, яке вони представляють, при цьому контролюючи цілу сукупність станів, дій, які пов’язані зі щоденною роботою з худобою. До цього входять загальні дані, дані вакцинацій, хвороб, фактичних і прогнозованих зборів молока, переходів на пасовиська, вимірювання стану свійської тварини, народження, продажу або придбання тощо.

Висновки. Наукова новизна роботи полягає в методі прогнозування стану худоби, що характеризується використанням дерев рішень, визначенням обчислення вхідних ознак, що дозволяє визначати обсяг удоїв худоби на наступний період. Проведене експериментальне дослідження підтвердило підвищення точності прогнозування стану худоби за використання запропонованого методу порівняно з методом опорних векторів.

Галузь використання. Аграрне господарство, тваринництво в частині обліку, відслідковування станів худоби господарства.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 114 pages, 26 figures, 3 tables, 3 appendixes, 25 sources.

CATTLE, FARM, MILK YIELD, FORECASTING, DECISION TREE, WEB APPLICATION.

The object of the work is the process of calculating the state of cattle.

The subject of the work is computer algorithms and software tools for tracking the condition of cattle.

The purpose of the work is to increase the accuracy of forecasting the condition of cattle by developing software tools for tracking the condition of cattle.

Materials, methods and technical means: Python language, Django framework, machine learning tools, in particular decision trees, MySQL system.

Results. The created cattle state tracking software is necessary for users to manage cattle data for the farm they represent, while controlling the entire set of conditions and actions associated with daily work with cattle. This includes general data, data on vaccinations, diseases, actual and predicted milk yield, walkings to pastures, measuring the state of cattle, birth, sale or purchase, etc.

Conclusions. The scientific novelty of the work lies in the method of forecasting the condition of cattle, characterized by the use of decision trees, the definition of the calculation of input features, which allows determining the volume of cattle milk yield for the next period. The conducted experimental study confirmed the increase in the accuracy of forecasting the state of cattle using the proposed method compared to the support vector method.

Field of application. Agriculture, cattle farming in terms of accounting, tracking the state of cattle on the farm.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

КСКВ – кореневе середньоквадратичне відхилення;

САП – середня абсолютна похибка;

СКП – середньоквадратична похибка.

ВСТУП

Робота з худобою господарства є доволі кропіткою, а у випадку наявності великої кількості худоби контроль за станом такої худоби стає проблематичним без використання програмних засобів.

Програмні засоби дозволяють розуміти поточний стан худоби, виходячи з історії спостережень за нею. З іншого боку ці ж дані можуть бути не тільки історією стосовно худоби, але і тими даними, на основі яких можна побудувати прогноз. Тобто розуміючи загальний стан худоби, робити прогноз щодо того, яку кількість молока вдасться зібрати в такої тварини. Це дозволяє планувати діяльність господарства, розуміючи, скільки продукції буде отримано.

У випадку проблем зі здоров’ям тварини необхідно мати можливість планувати подальшу роботу, орієнтуючись на близькі до подальших фактичних результатів. Тому інтеграція інструментів прогнозування зокрема удоїв молока худоби дозволить мати комплексний програмний засіб для відслідковування стану худоби господарства, стаючи основою для подальшого планування роботи.

Цим і пояснюється актуальність розробки програми відслідковування стану худоби, що розробляється в цій роботі для збільшення точності прогнозування стану худоби.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд вебзастосунків відслідковування стану худоби

FarmKeep [1] (рис. 1.1) є застосунком, який може використовуватися для відслідковування стану худоби. Він представлений як застосунок для Android та iOS. Доступний в безкоштовній та платній версіях. Тільки платна версія передбачає можливість ведення декількох господарств, внесення даних вакцинації тварин, дій тощо.

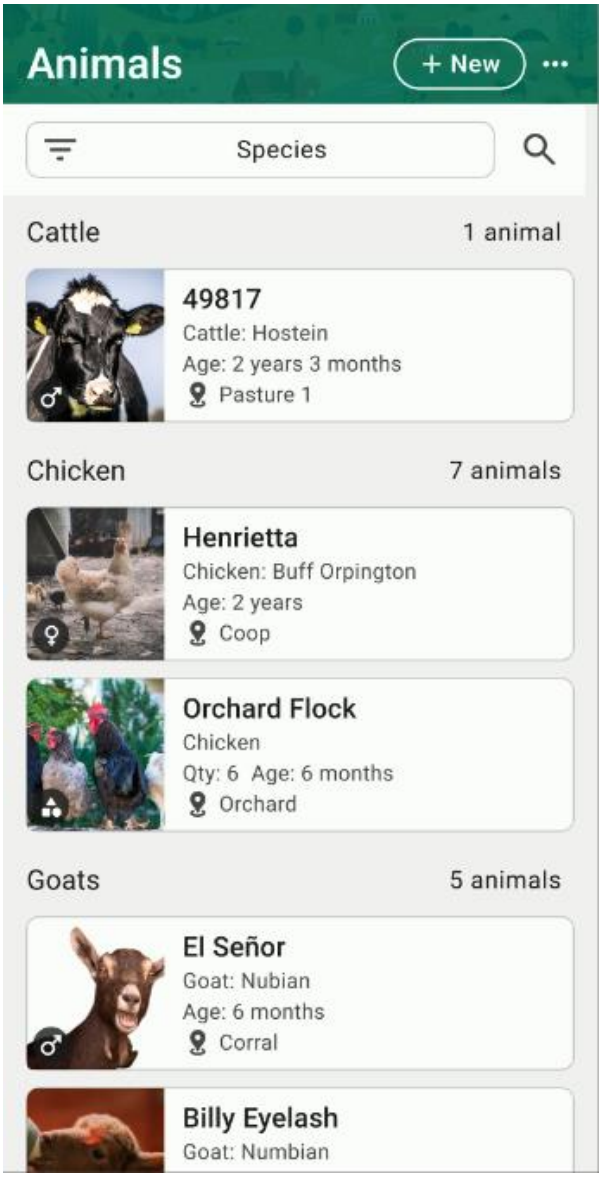

Рисунок 1.1 – Застосунок FarmKeep [2]

FarmKeep дозволяє відслідковувати породи, вік худоби, народжуваність худоби, вносити, відслідковувати щоденні дії, пов’язані з годуванням худоби, відслідковувати зміни стану здоров’я, вносити та відслідковувати вакцинації.

Cattlytics [3] (рис. 1.2) доступний для користувачів не тільки як застосунок для Android, iOS, але як і вебзастосунок. Він є платним для використання, а безкоштовно може використовуватися тільки після запиту пробної версії протягом 15 днів.

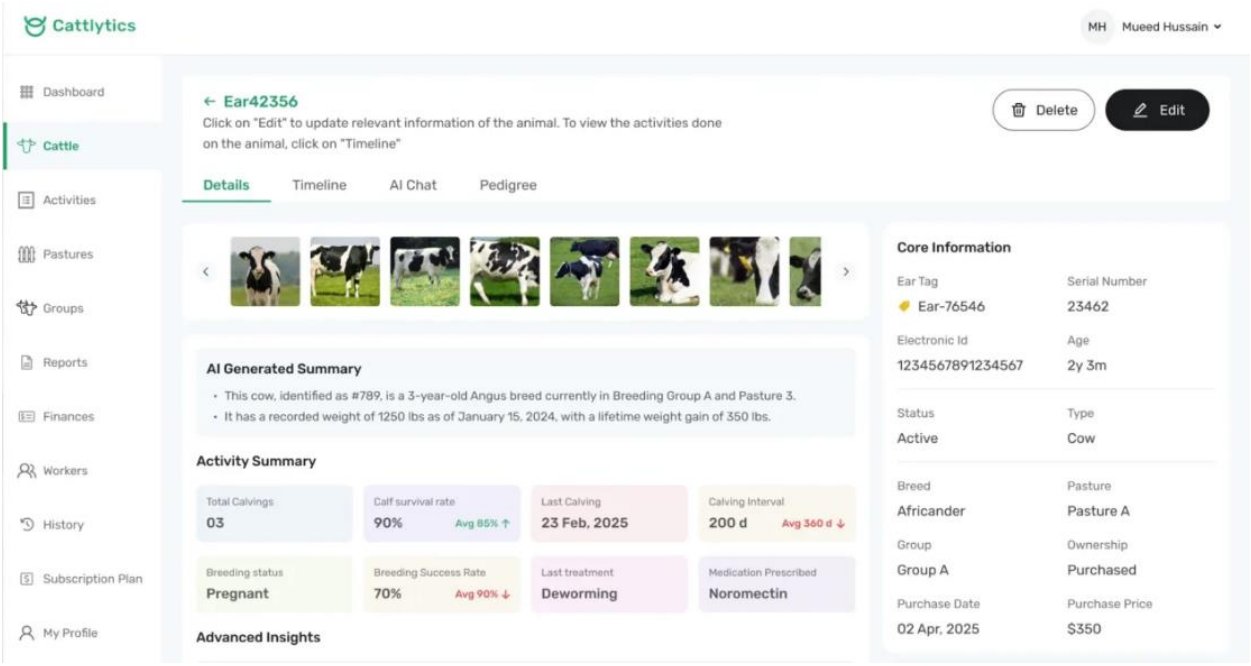

Рисунок 1.2 – Застосунок Cattlytics [4]

Cattlytics дозволяє за худобою вносити і відслідковувати загальні дані, удої, годування, стан здоров’я тварин, народження. Cattlytics, як і FarmKeep, не має засобів прогнозування удоїв, дозволяючи тільки робити висновки користувачу на основі всієї сукупності наявних даних про стан тварини. Додатковою можливістю Cattlytics, FarmKeep є можливість інтеграції з пристроями відстеження.

1.2 Порівняння аналогів розробки вебзастосунку відслідковування стану худоби

Порівняння аналогів розробки вебзастосунку відслідковування стану худоби, що розробляється в цій роботі та має назву CattleMoni, відбувалось з FarmKeep і Cattlytics (табл. 1.1).

Таблиця 1.1 – Порівняння аналогів вебзастосунку відслідковування стану худоби

Показник	FarmKeep	Cattlytics	CattleMoni
Внесення та відслідковування народжень	Так	Так	Так
Внесення та відслідковування годування худоби	Так	Так	Так
Внесення та відслідковування стану худоби	Так	Так	Так
Внесення та відслідковування виконаних дій з худобою	Так	Так	Так
Інтеграція з пристроями відстеження	Так	Так	Ні
Безкоштовний доступ	Обмежені функції, зокрема неможливе внесення вакцинувань	Тільки демоверсія	Повністю
Прогнозування удоїв худоби	Ні	Ні	Так

Вебзастосунок відслідковування стану худоби CattleMoni, що розробляється в роботі, в якості ключової відмінності від аналогів має наявність функції прогнозування удоїв худоби, окрім того це повністю безкоштовний застосунок, в той час як аналоги не дозволяють взагалі або повноцінно працювати безкоштовно. Перевагою аналогів є можливість інтеграції цих застосунків з пристроями відстеження, що дозволяє тоді збирати дані про місцезнаходження худоби. Це може бути подальшим розвитком програми, що розробляється, проте в цій роботі реалізація таких можливостей не запланована.

1.3 Висновки за розділом 1

Метою роботи є збільшення точності прогнозування стану худоби шляхом розробки програмних засобів відслідковування стану худоби. Завдання, які мають забезпечити досягнення мети:

– аналіз програм відслідковування стану худоби;

– розробка методу прогнозування стану худоби;

– проєктування програми відслідковування стану худоби;

– реалізація програми відслідковування стану худоби;

– експериментальне дослідження прогнозування удою худоби.

Відмінністю розробки від аналогів (FarmKeep, Cattlytics) є наявність функції прогнозування удоїв худоби та можливість повноцінного безкоштовного використання.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Задача прогнозування обсягу удоїв худоби

Задача прогнозування обсягу удоїв худоби є задачею, в якій потрібно визначити вид функції залежності вихідної ознаки від вхідних ознак на основі наявних даних про минулі спостереження за станом худоби господарства. Це встановлює залежність між вхідними та вихідною ознакою.

Вихідною ознакою є обсяг молока, яку в середньому з цієї свійської тварини на добу буде надоєно протягом наступного періоду, який є періодом прогнозування. Це може бути тиждень або, наприклад, місяць.

Вхідними ознаками для задачі прогнозування обсягу удоїв худоби зокрема є:

– кількість народженої худоби свійською твариною;

– порода худоби;

– кількість послідовних днів, протягом якої дає молоко ця свійська тварина;

– кількість води, яку в середньому споживає на день ця тварина;

– середня температура повітря на прогнозований період часу;

– відсоток вологості повітря на прогнозований період часу;

– вага їжі, яку споживає на день в середньому свійська тварина;

– сезон, на який відбувається прогнозування;

– середня кількість кілометрів, яку долає на день свійська тварина;

– поточна повна кількість місяців віку свійської тварини;

– регіон розміщення породи худоби;

– оцінка приміщення, в якому знаходиться худоба (від 1 до 5);

– середня кількість годин відпочинку свійської тварини на добу;

– середня кількість годин випасання свійської тварини на добу;

– країна знаходження свійської тварини;

– тип годування свійської тварини;

– клімат знаходження свійської тварини;

– середня кількість годин проміжку між удоями;

– оцінка стану тіла худоби;

– частота годування свійської тварини на добу;

– система ведення господарства;

– стадія лактації свійської тварини;

– середня кількість в літрах удою молока на добу за попередній аналогічний прогнозуванню період;

– середня тривалість румінації свійської тварини в годинах на добу;

– наявність вакцини проти сказу;

– наявність вакцини проти сибірської виразки;

– наявність вакцини проти бруцельозу;

– наявність вакцини проти ящура;

– наявність вакцини проти герпесу;

– наявність вакцини проти вірусної діареї;

– наявність вакцини проти маститу;

– наявність вакцини проти інфекційного ринотрахеїту.

2.2 Метод прогнозування стану худоби

Метод прогнозування стану худоби передбачає на першому етапі збір даних, що є спостереженнями за худобою та встановлення періоду прогнозування. Кожен екземпляр майбутньої вибірки даних повинен узагальнювати стан худоби на момент прогнозування, а майбутній стан має бути зафіксований у вигляді вихідної ознаки. Сукупність вхідних ознак вибірки даних може включати ознаки з підрозділу 2.1.

На наступному етапі реалізується оброблення даних зі сформованої вибірки даних. Зокрема це може реалізовуватись на прикладі структури з підрозділу 2.1 наступним чином:

– кількість народженої худоби свійською твариною, що представлена цілим числом, а у випадку відсутності значення має бути встановлено на рівні 0;

– порода худоби, яка має бути перетворена до категоріального значення і зведена до цілого числа, при цьому у випадку пропущеного значення за ознакою використовується найбільш розповсюджена порода худоби господарства (ознака може бути прибрана, якщо дані зібрані за однією єдиною породою, що аналогічно для всіх майбутніх даних);

– кількість послідовних днів, протягом якої дає молоко ця свійська тварина, що визначається кількістю днів, що минула з часу, коли худоба востаннє не давала молоко, і має бути представлена цілим числом (у випадку відсутності значення варто прибрати спостереження з вибірки);

– кількість води, яку в середньому споживає на день ця тварина і яка має бути представлена числом з плаваючою крапкою (має бути встановлено стандартний обсяг, що використовується як значення за замовчуванням у випадку пропусків);

– середня добова температура повітря на прогнозований період часу, яка має бути представлена числом з плаваючою крапкою;

– відсоток вологості повітря на прогнозований період часу, який має бути представлена числом з плаваючою крапкою;

– вага їжі, яку споживає на день в середньому свійська тварина, представлена числом з плаваючою крапкою, що у випадку пропуску значення має бути замінена стандартним обсягом як значення за замовчуванням;

– сезон, на який відбувається прогнозування;

– середня кількість кілометрів, яку долає на день свійська тварина, представлена числом з плаваючою крапкою, що у випадку пропуску значення має бути замінена стандартним обсягом як значення за замовчуванням;

– поточна повна кількість місяців віку свійської тварини (у випадку відсутності значення варто прибрати спостереження з вибірки);

– регіон розміщення породи худоби, яка має бути перетворена до категоріального значення і зведена до цілого числа, при цьому у випадку пропущеного значення за ознакою має бути уточнено довідкові дані (ознака може бути прибрана, якщо дані зібрані за однією єдиною породою худоби, що аналогічно для всіх майбутніх даних);

– оцінка приміщення, в якому знаходиться худоба (від 1 до 5), представлена числом з плаваючою крапкою, що у випадку пропуску значення має бути замінена стандартним значенням, що відповідає загальному стану приміщень (ознака може бути прибрана, якщо дані зібрані за одним єдиним приміщенням, що аналогічно для всіх майбутніх даних);

– середня кількість годин відпочинку свійської тварини на добу, представлена числом з плаваючою крапкою, що у випадку пропуску значення має бути замінена стандартним обсягом як значення за замовчуванням;

– середня кількість годин випасання свійської тварини на добу, представлена числом з плаваючою крапкою, що у випадку пропуску значення має бути замінена стандартним обсягом як значення за замовчуванням;

– країна знаходження свійської тварини, яка має бути перетворена до категоріального значення і зведена до цілого числа (ознака може бути прибрана, якщо дані зібрані за одним єдиним господарством);

– тип годування свійської тварини, який має бути перетворений до категоріального значення і зведений до цілого числа, при цьому у випадку пропущеного значення використовується стандартний заданий тип годування;

– клімат знаходження свійської тварини, який має бути перетворений до категоріального значення і зведений до цілого числа (ознака може бути прибрана, якщо дані зібрані за одним єдиним господарством);

– середня кількість годин проміжку між удоями, представлена числом з плаваючою крапкою, що у випадку пропуску значення має бути замінена стандартним обсягом як значення за замовчуванням;

– оцінка стану тіла худоби, представлена числом з плаваючою крапкою в діапазоні від 1 до 5, що у випадку пропуску значення має бути замінена стандартним числом як значення за замовчуванням;

– частота годування свійської тварини на добу, представлена числом з плаваючою крапкою, що у випадку пропуску значення має бути замінена стандартним обсягом як значення за замовчуванням;

– система ведення господарства, яка має бути перетворена до категоріального значення і зведена до цілого числа, при цьому у випадку пропущеного значення за ознакою має бути використано стандартні дані (ознака може бути прибрана, якщо дані зібрані за одним єдиним господарством з однією єдиною системою, що аналогічно для всіх майбутніх даних);

– стадія лактації свійської тварини, яка має бути перетворена до категоріального значення і зведена до цілого числа (у випадку відсутності значення варто прибрати спостереження з вибірки);

– середня кількість в літрах удою молока на добу за попередній аналогічний прогнозуванню період, представлена числом з плаваючою крапкою (у випадку відсутності значення варто прибрати спостереження з вибірки);

– середня тривалість румінації свійської тварини в годинах на добу, представлена числом з плаваючою крапкою, що у випадку пропуску значення має бути замінена стандартним обсягом як значення за замовчуванням;

– наявність вакцини проти сказу зі встановленням значення 1, якщо вакцина була зроблена, та 0, якщо ні (у випадку пропуску значення використовується 0);

– наявність вакцини проти сибірської виразки зі встановленням значення 1, якщо вакцина була зроблена, та 0, якщо ні (у випадку пропуску значення використовується 0);

– наявність вакцини проти бруцельозу зі встановленням значення 1, якщо вакцина була зроблена, та 0, якщо ні (у випадку пропуску значення використовується 0);

– наявність вакцини проти ящура зі встановленням значення 1, якщо вакцина була зроблена, та 0, якщо ні (у випадку пропуску значення використовується 0);

– наявність вакцини проти герпесу зі встановленням значення 1, якщо вакцина була зроблена, та 0, якщо ні (у випадку пропуску значення використовується 0);

– наявність вакцини проти вірусної діареї зі встановленням значення 1, якщо вакцина була зроблена, та 0, якщо ні (у випадку пропуску значення використовується 0);

– наявність вакцини проти маститу зі встановленням значення 1, якщо вакцина була зроблена, та 0, якщо ні (у випадку пропуску значення використовується 0);

– наявність вакцини проти інфекційного ринотрахеїту зі встановленням значення 1, якщо вакцина була зроблена, та 0, якщо ні (у випадку пропуску значення використовується 0);

– середня кількість в літрах удою молока на добу за наступний період (у випадку відсутності значення варто прибрати спостереження з вибірки).

Остання ознака є вихідною. Значення вихідної ознаки нормуються.

Окрім того всі дублювання даних мають бути вилучені з вибірки.

На наступному етапі створюється модель на основі дерева рішень [5]-[7] для прогнозування обсягу удоїв худоби.

Далі реалізується навчання такої моделі прогнозування обсягу удоїв худоби за допомогою створеної на перших 2 етапах вибірки даних.

На останньому етапі, який на час життя навченої моделі прогнозування обсягу удоїв худоби має повторюватись при появі необхідності нового прогнозування, виконується кодування даних спостереження за худобою на основі сукупності вхідних ознак у спосіб, описаний для другого етапу. Тоді

для виконаного кодування реалізується прогнозування обсягу удоїв худоби навченою моделлю (рис. 2.1).

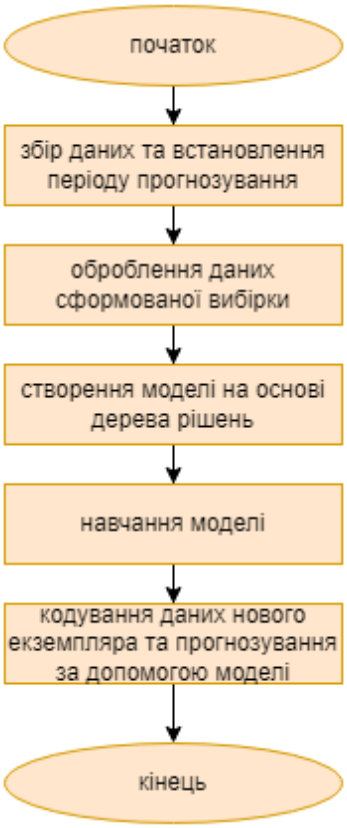

Рисунок 2.1 – Схема роботи методу прогнозування стану худоби

2.3 Висновки за розділом 2

Описано задачу прогнозування обсягу удоїв худоби та метод, який представлено для її розв’язання. Наукова новизна роботи полягає в методі прогнозування стану худоби, що характеризується використанням дерев рішень, визначенням обчислення вхідних ознак, що дозволяє визначати обсяг удоїв худоби на наступний період.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір мови та засобів розробки

Основне порівняння стосовно вибору мови для реалізації програми відслідковування стану худоби зведено до Python [8]-[13] і Ruby [14]-[16], зважаючи на активність засобів веброзробки, у вигляді якої програма має бути розроблена, наявність фреймворків відповідно Flask і Django [17]-[19], який має більше вбудованих засобів, та Ruby on Rails (табл. 3.1). Вибір на користь Python, Django зроблений через те, що мова програмування сприяє простоті роботи з даними, підтримці машинного навчання за рахунок наявності готових бібліотек, широкої підтримки спільноти. Для організації збереження даних у вигляді бази даних вирішено обрати систему MySQL.

Таблиця 3.1 – Вибір мови для реалізації програми відслідковування стану худоби

	Python	Ruby
Наявність фреймворку для підтримки веброзробки	Так, Django, Flask	Так, Ruby on Rails
Коло інструментів для машинного навчання і роботи з даними	Широке	Вузьке
Підтримка	Велика спільнота	Вузьконаправлена
Можливості навчання	Високі	Високі

3.2 Моделювання роботи користувачів

Усі користувачі програми відслідковування стану худоби мають одну й ту саму роль. Фактично є тільки окремо виділена група користувачів-адміністраторів, які повинні мати можливість працювати з програмою через

адміністративну панель. Адміністративна панель включає всі сценарії додавання, редагування, вилучення даних за кожною таблицею бази даних. Однак такі сценарії є повністю типовими, не вимагають окремої розробки коду, інтерфейсів вебсторінок, тому вони не виділялись окремо. Відповідно під час моделювання використано стандартного користувача як єдину групу користувачів.

Слід також зазначити, що користувач без авторизації з програмою відслідковування стану худоби працювати не може, оскільки він обов’язково повинен виступати в ролі, зв’язаній з певним господарством для того, щоб мати можливість виконувати роботу з даними в програмі, оскільки всі вони також пов’язані з кожним господарством, а відповідно необхідно забезпечити безпеку таких даних.

Основний потік сценаріїв роботи користувача з програмою відслідковування стану худоби включає:

– перегляд даних власного господарства;

– авторизація;

– перегляд користувачів, які представляють господарство;

– надання права працювати від імені господарства користувачу;

– додавання приміщення до господарства;

– перегляд всіх приміщень господарства;

– перегляд всієї худоби, зареєстрованої за господарством;

– реєстрація нової худоби господарством;

– реєстрація продажу худоби;

– перегляд даних приміщення господарства;

– реєстрація придбання існуючої в програмі худоби;

– перегляд всієї придбаної господарством худоби;

– перегляд всієї проданої господарством худоби;

– перегляд поголів’я народженої худоби в господарстві;

– перегляд поголів’я народженої твариною худоби;

– перегляд динаміки народження худоби за господарством;

– внесення удою худоби за твариною;

– внесення удою худоби за породою тварин;

– внесення удою худоби за приміщенням господарства;

– перегляд динаміки удоїв худоби загалом за господарством;

– перегляд динаміки удоїв худоби заданої породи за господарством;

– перегляд динаміки удоїв конкретної тварини;

– перегляд історії удоїв худоби за приміщенням господарства;

– перегляд прогнозувань удою худоби;

– прогнозування удою худоби;

– встановлення базового варіанта годування худоби;

– встановлення параметрів годування худоби;

– внесення годування худоби за твариною;

– внесення годування худоби за породою тварин;

– внесення годування худоби за приміщенням господарства;

– перегляд історії годування худоби за приміщенням господарства;

– перегляд історії годування худоби за твариною;

– перегляд історії зміни варіантів годування худоби за твариною;

– перегляд історії зміни параметрів годування худоби за твариною;

– перегляд історії зміни варіантів годування худоби в господарстві за породою;

– перегляд історії зміни параметрів годування худоби в господарстві за породою;

– перегляд історії зміни обсягів годування худоби в господарстві за варіантом годування;

– внесення дій над худобою за твариною;

– внесення дій над худобою за породою тварин;

– внесення дій над худобою за приміщенням господарства;

– визначення завершення дії за худобою;

– перегляд дій над худобою за твариною;

– перегляд дій над худобою за приміщенням господарства;

– перегляд активності дій над худобою за господарством;

– внесення вимірювань стану худоби за твариною;

– внесення вимірювань стану худоби за приміщенням господарства;

– внесення вимірювань стану худоби за породою;

– перегляд історії зміни стану худоби за твариною;

– перегляд динаміки зміни стану худоби за приміщенням господарства;

– перегляд динаміки зміни стану худоби за породою;

– внесення вакцинації худоби за твариною;

– внесення вакцинації худоби за приміщенням господарства;

– внесення вакцинації худоби за породою;

– перегляд історії вакцинації худоби за твариною;

– перегляд стану вакцинації худоби за приміщенням господарства;

– перегляд стану вакцинації худоби за породою;

– внесення хвороби худоби;

– визначення завершення хвороби худоби;

– перегляд історії захворювання худоби за твариною;

– перегляд історії захворювання худоби за приміщенням господарства;

– перегляд історії захворювання худоби за породою;

– перегляд динаміки захворювання худоби за господарством;

– перегляд динаміки захворювання худоби за приміщенням господарства;

– перегляд динаміки захворювання худоби за породою;

– пошук худоби за власним господарством;

– пошук худоби за іншим господарством;

– перегляд поданих заявок придбань худоби господарства;

– перегляд власних поданих заявок придбань худоби;

– внесення погодних умов за господарством;

– перегляд внесених погодних умов за господарством;

– підтвердження продажу худоби (рис. 3.1).

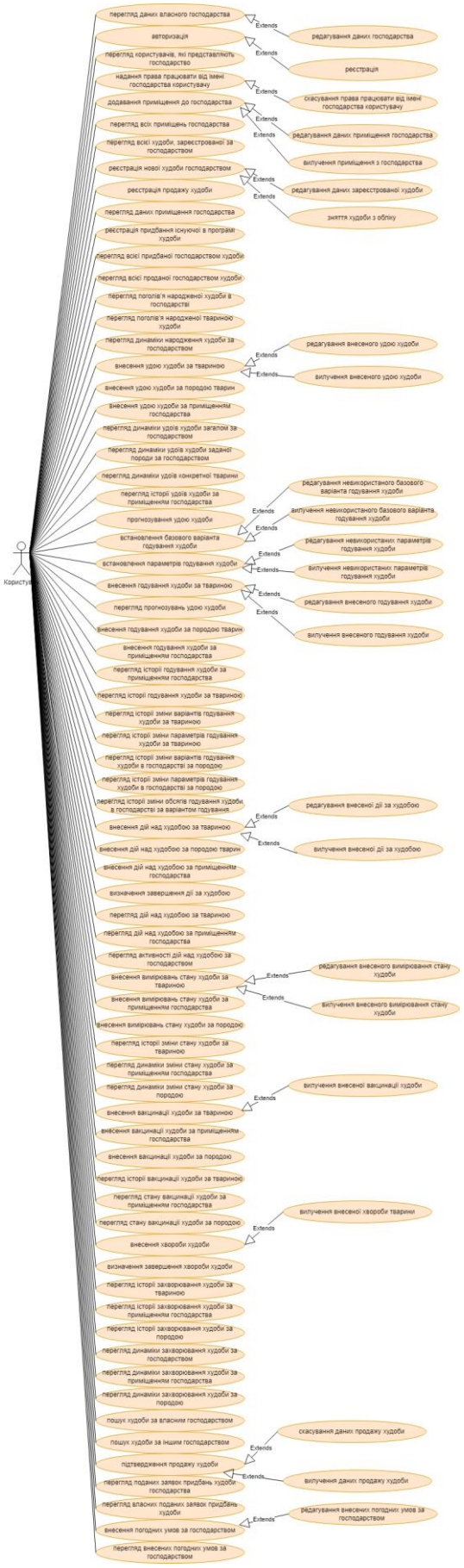

Рисунок 3.1 – Діаграма прецедентів для користувача програми відслідковування стану худоби

Альтернативними сценаріями є:

– реєстрація для авторизації, коли акаунт у користувача відсутній;

– скасування права працювати від імені господарства користувачу для надання права працювати від імені господарства користувачу, коли користувач вже працює від імені господарства, проте таке право має бути скасовано;

– редагування даних господарства для перегляду даних власного господарства, коли дані господарства містять поточні некоректності;

– редагування даних приміщення господарства для додавання приміщення до господарства, коли приміщення господарства вже додано, але його дані некоректні на поточний момент;

– вилучення приміщення з господарства для додавання приміщення до господарства, коли приміщення господарства вже додано, але воно вже відсутнє в реальності;

– редагування даних зареєстрованої худоби для реєстрації нової худоби господарством, коли реєстрація худоби виконана, але дані про худобу містять некоректні на поточний момент;

– зняття худоби з обліку для реєстрації нової худоби господарством, коли реєстрація худоби виконана, але ця худоба вже не присутня в цьому господарстві або взагалі, при цьому вона і не продана іншому господарству з програми;

– редагування внесеного удою худоби для внесення удою худоби за твариною, коли дані про удій худоби вже внесені, але неточно;

– вилучення внесеного удою худоби для внесення удою худоби за твариною, коли дані про удій худоби вже внесені, але помилково;

– редагування внесеного годування худоби для внесення годування худоби за твариною, коли таке годування худоби вже було внесено, але містить неточності;

– вилучення внесеного годування худоби для внесення годування худоби за твариною, коли таке годування худоби вже було внесено, але помилково;

– редагування внесеної дії за худобою для внесення дій над худобою за твариною, коли дані дій за худобою були внесені, але містять неточності;

– вилучення внесеної дії за худобою для внесення дій над худобою за твариною, коли дані дій за худобою були внесені, але помилково;

– вилучення внесеної хвороби тварини для внесення хвороби худоби, коли хвороба за твариною була внесена, але помилково;

– вилучення внесеної вакцинації худоби для внесення вакцинації худоби за твариною, коли вакцинація худоби була внесена, але помилково;

– скасування даних продажу худоби для підтвердження продажу худоби, коли початкові дані про продаж худоби були внесені однією стороною, але друга вважає це помилкою;

– вилучення даних продажу худоби для підтвердження продажу худоби, коли початкові дані про продаж худоби були внесені, але сам ініціатор визнає їх помилку;

– редагування внесених погодних умов за господарством для внесення погодних умов за господарством, коли погодні умови за господарством були внесені, але містять некоректності;

– редагування внесеного вимірювання стану худоби для внесення вимірювань стану худоби за твариною, коли вимірювання стану худоби були внесені, але містять неточності;

– вилучення внесеного вимірювання стану худоби для внесення вимірювань стану худоби за твариною, коли вимірювання стану худоби були внесені, але помилково;

– вилучення невикористаного базового варіанта годування худоби для встановлення базового варіанта годування худоби, коли базовий варіант годування худоби був встановлений, але помилково;

– редагування невикористаного базового варіанта годування худоби для встановлення базового варіанта годування худоби, коли базовий варіант годування худоби був встановлений, але містить неточності;

– редагування невикористаних параметрів годування худоби для встановлення параметрів годування худоби, коли параметри годування худоби були встановлені, але містять неточності;

– вилучення невикористаних параметрів годування худоби для встановлення параметрів годування худоби, коли параметри годування худоби були встановлені, але помилково.

3.3 Проєктування структури бази даних

Для визначення структури бази даних було виділено основні сутності предметної області та відношення між ними.

Звісно, що худоба є однією з основних сутностей предметної області. Проте зважаючи на особливості використання цього терміну в українській мові, в подальшому термін худоба використаний для позначення сукупності деяких свійських тварин, а окремий екземпляр позначається терміном свійська тварина.

Таблиця Cattleregion містить світові регіони розповсюдження худоби, включаючи стовпці:

– id – ідентифікатор світового регіону розповсюдження худоби;

– cregionname – назва світового регіону розповсюдження худоби.

Таблиця Cattlecountry містить країни, в яких може вирощуватись худоба, в яких можуть знаходитись господарства, включаючи стовпці:

– id – ідентифікатор країни;

– ccouname – назва країни;

– cregion – світовий регіон розповсюдження худоби, де знаходиться ця країна, що визначається зовнішнім ключем з таблиці Cattleregion, формуючи відношення «один-до-багатьох».

Таблиця Cattlebreed містить породи худоби, включаючи стовпці:

– id – ідентифікатор породи худоби;

– cbreedname – назва породи худоби;

– cbreedinfo – текстова інформація стосовно особливостей вирощування цієї породи худоби;

– mainregion – основний світовий регіон, де розповсюджена худоба цієї породи, що визначається зовнішнім ключем з таблиці Cattleregion, формуючи відношення «один-до-багатьох»;

– maincountry – основна країна, де вирощується худоба цієї породи, що визначається зовнішнім ключем з таблиці Cattlecountry, формуючи відношення «один-до-багатьох»;

– basemanagement – система ведення господарства, яка зазвичай має застосовуватися для такої породи худоби;

– cbreedphoto – фотографія худоби цієї породи.

Необхідність збереження і країна, і світового регіону для породи худоби, не зважаючи на відношення кожної країни до певного світового регіону вирощування худоби, пояснюється тим, що для породи худоби може бути не задана країна, а тільки регіон.

Таблиця Cattle містить дані конкретних тварин, включаючи стовпці:

– id – ідентифікатор свійської тварини;

– cattlebreed – порода худоби, до якої ця свійська тварина належить, що визначається зовнішнім ключем з таблиці Cattlebreed, формуючи відношення «один-до-багатьох»;

– inventnumber – інвентарний номер цієї свійської тварини серед всієї худоби господарства: він не використовується в якості первинного ключа в таблиці тому, що може бути не заданий, якщо господарство має невелику кількість тварин, не веде їх ідентифікацію власним чином, окрім того він є інвентарним номером серед всіх тварин одного господарства, а не всіх, які працюють з програмою;

– cattlealias – кличка цієї свійської тварини, якщо господарство встановлює їх тваринам і ця її має;

– barnlocation – приміщення господарства, в якому знаходиться ця свійська тварина, що визначається зовнішнім ключем з таблиці Barn, формуючи відношення «один-до-багатьох»;

– fullbirthdate – дата народження цієї свійської тварини;

– birthtimenumber – кількість худоби, яку народила ця свійська тварина (за час спостережень за нею);

– lactationstage – фаза лактації, в якій на даний момент знаходиться ця свійська тварина: роздоювання, стійкий удій, зниження, сухостійний період;

– state – стан, в якому на даний момент знаходиться ця свійська тварина;

– cattlephoto – фотографія цієї свійської тварини.

Таблиця Cattlebirth містить дані народження худоби в господарстві, включаючи стовпці:

– id – ідентифікатор народження худоби;

– cattlemother – свійська тварина, яка народила, що визначається зовнішнім ключем з таблиці Cattle, формуючи відношення «один-до-багатьох»;

– cattlechild – свійська тварина, яку було народжено, що визначається зовнішнім ключем з таблиці Cattle, формуючи відношення «один-до-одного», оскільки худоба народжується тільки один раз;

– birthinfo – додаткова інформація стосовно народження цієї худоби.

Таблиця Cattlemilking містить дані молочної продуктивності худоби, тобто проведених удоїв молока, включаючи стовпці:

– id – ідентифікатор запису молочної продуктивності худоби;

– cattle – свійська тварина, удій молока якої було проведено, що визначається зовнішнім ключем з таблиці Cattle, формуючи відношення «один-до-багатьох»;

– milkvolume – обсяг молока, який було отримано під час цього удою цієї свійської тварини;

– milkingtime – час початку цього удою цієї свійської тварини.

Таблиця Cattleprediction містить прогнозування удоїв молока худоби, включаючи стовпці:

– id – ідентифікатор виконаного прогнозування удою молока цієї худоби;

– cattle – свійська тварина, прогнозування удою молока якої було виконано, що визначається зовнішнім ключем з таблиці Cattle, формуючи відношення «один-до-багатьох»;

– predictedmilkvolume – обсяг удою молока, який буде в середньому отримано за наступний період (тиждень) для цієї свійської тварини за результатами прогнозування;

– predictiontime – час виконання прогнозування удою молока цієї худоби;

– datastart – час, починаючи з якого дані про стан, удій цієї свійської тварини були використані при прогнозуванні для усереднення в якості вхідних даних моделі;

– dataend – час, до якого дані про стан, удій цієї свійської тварини були використані при прогнозуванні для усереднення в якості вхідних даних моделі.

Таблиця Cattlefeedbasetype містить типові варіанти харчування худоби, включаючи стовпці:

– id – ідентифікатор типового варіанта харчування худоби;

– feedbtname – назва типового варіанта харчування худоби (наприклад, сіно).

Таблиця Cattlefeedtype містить варіанти харчування, які застосовуються до конкретних свійських тварин, включаючи стовпці:

– id – ідентифікатор варіанта харчування, який застосовується до цієї свійської тварини;

– cattle – свійська тварина, до якої застосовується цей варіант харчування, що визначається зовнішнім ключем з таблиці Cattle, формуючи відношення «один-до-багатьох»;

– cattlefeedbasetype – типовий варіант харчування, який застосовується для цієї свійської тварини у цей період часу, що визначається зовнішнім ключем з таблиці Cattlefeedbasetype, формуючи відношення «один-до-багатьох»;

– feedtypefixtimestart – час, з якого почалось застосування цього варіанта харчування для цієї свійської тварини;

– feedtypefixtimeend – час, до якого тривало застосування цього варіанта харчування для цієї свійської тварини (якщо не задано значення, то такий варіант харчування є поточним).

Таблиця Cattlefeedbase містить деталізацію варіантів харчування, які застосовуються до конкретних свійських тварин, включаючи стовпці:

– id – ідентифікатор деталізації варіанта харчування, який застосовується до цієї свійської тварини;

– cattlefeedtype – варіант харчування, який застосовується до цієї свійської тварини та який деталізовано таким чином, що визначається зовнішнім ключем з таблиці Cattlefeedtype, формуючи відношення «один-до-багатьох»;

– feedquantitykg – обсяг харчування за цим варіантом, що застосовується для цієї свійської тварини, заданий в кілограмах;

– waterintake – обсяг води, який використовується для пиття цієї свійської тварини в літрах за час між харчуваннями;

– feedfixstart – час, з якого діє цей обсяг (деталізація) варіанта харчування цієї свійської тварини;

– feedfixend – час, до якого діє цей обсяг (деталізація) варіанта харчування цієї свійської тварини (якщо не задано значення, то така деталізація цього варіанта харчування є поточною).

Таблиця Cattlefeed містить випадки годування свійських тварин, включаючи стовпці:

– id – ідентифікатор цього годування цієї свійської тварини;

– cattlefeedbase – деталізація варіанта харчування свійської тварини, яку було застосовано під час цього годування, що визначається зовнішнім ключем з таблиці Cattlefeedbase, формуючи відношення «один-до-багатьох»;

– feedtime – час початку цього годування цієї свійської тварини.

Таблиця Cattleactivity містить дії (відпочинок, випас, перехід на пасовисько), які виконувались стосовно свійських тварин, включаючи стовпці:

– id – ідентифікатор цієї дії, яка виконувалась стосовно цієї свійської тварини;

– cattle – свійська тварина, до якої було виконано цю дію, що визначається зовнішнім ключем з таблиці Cattle, формуючи відношення «один-до-багатьох»;

– activitystarttime – час початку реалізації цієї дії до цієї свійської тварини;

– activityendtime – час завершення реалізації цієї дії до цієї свійської тварини;

– activitytype – тип дії, яку було виконано до цієї свійської тварини (відпочинок, випас, перехід на пасовисько).

Перехід тварин на пасовисько був деталізований окремо, оскільки він передбачає збереження дистанції, яку подолала худоба під час переходу.

Таблиця Cattlewalking містить дані про перехід тварин на пасовисько, включаючи стовпці:

– id – ідентифікатор переходу тварини на пасовисько або іншого переходу тварини, який передбачав долання певної дистанції;

– cattleactivity – дія, якою було описано всю стандартну частину цього переходу тварини на пасовисько, що визначається зовнішнім ключем з таблиці Cattleactivity, формуючи відношення «один-до-одного»;

– walkingdistancekm – відстань в кілометрах, яку подолала ця свійська тварина під час такого переходу за цією дією.

Таблиця Cattlemeasurement містить вимірювання стану (температури, серцебиття, частоти дихання, стану тіла) свійських тварин, включаючи стовпці:

– id – ідентифікатор цього вимірювання стану цієї свійської тварини;

– cattle – свійська тварина, вимірювання стану якої було виконано, що визначається зовнішнім ключем з таблиці Cattle, формуючи відношення «один-до-багатьох»;

– measuretime – час, коли це вимірювання стану цієї свійської тварини було виконано;

– cattlemeasurevalue – значення показника, яке було виміряно для цієї світської тварини у цю мить;

– cattlemeasuretype – тип показника, який було виміряно для цієї світської тварини у цю мить: температура в градусах, рівень серцебиття в кількості ударів серця на хвилину, частота дихання в кількості подихів на хвилину, стан тіла за оцінкою від 1 до 5).

Таблиця Ambientweather містить спостереження за погодними умовами навколишнього середовища, включаючи стовпці:

– id – ідентифікатор спостереження за погодними умовами навколишнього середовища;

– household – господарство, де таке спостереження відбувалось, що визначається зовнішнім ключем з таблиці Household, формуючи відношення «один-до-багатьох»;

– ambienttime – час спостереження за погодними умовами навколишнього середовища;

– ambienttemp – температура навколишнього середовища в цей час в цьому населеному пункті;

– ambienthumidity – вологість повітря навколишнього середовища в цей час в цьому населеному пункті.

Таблиця Locationplace містить місця (населені пункти), де можуть знаходитись господарства, відбуватися спостереження за погодними умовами навколишнього середовища, включаючи стовпці:

– id – ідентифікатор місця (населеного пункту);

– cattlecoutry – країна, де це місце (населений пункт) знаходиться, що визначається зовнішнім ключем з таблиці Cattlecountry, формуючи відношення «один-до-багатьох»;

– placename – назва місця (населеного пункту).

Таблиця Household містить господарства, які використовуючи програму, володіють худобою, включаючи стовпці:

– id – ідентифікатор господарства, яке володіє худобою;

– locationplace – місце (населений пункт), де знаходиться це господарство, що визначається зовнішнім ключем з таблиці Locationplace, формуючи відношення «один-до-багатьох»;

– locationdetail – деталізація місця розташування господарства в цьому населеному пункті або біля нього у вигляді текстового опису (можливо конкретна адреса);

– locationlat – широта місця розташування господарства (для відображення на мапі);

– locationlong – довгота місця розташування господарства (для відображення на мапі);

– householdtitle – назва господарства;

– hhphoto – фотографія господарства.

Таблиця Householduser містить облікові записи, які можуть вносити дані стосовно цього господарства, включаючи стовпці:

– id – ідентифікатор права облікового запису працювати з даними господарства;

– household – господарства, з яким може працювати користувач з цим обліковим записом, що визначається зовнішнім ключем з таблиці Household, формуючи відношення «один-до-багатьох»;

– user – обліковий запис користувача, який може працювати з даними цього господарства, що визначається зовнішнім ключем зі стандартної таблиці User, формуючи відношення «один-до-одного», оскільки користувач може представляти тільки одне господарство в програмі одночасно.

Таблиця Barn містить приміщення господарства, в яких знаходиться худоба, включаючи стовпці:

– id – ідентифікатор приміщення господарства, в якому знаходиться худоба;

– household – господарство, яке володіє цим приміщенням, де може знаходитись худоба, що визначається зовнішнім ключем з таблиці Household, формуючи відношення «один-до-багатьох»;

– barnname – назва приміщення, якщо воно має конкретну назву, або певна позначка для текстової ідентифікації;

– barnscore – оцінка стану приміщення стосовно збереження в ньому худоби від 1 до 5;

– cattlemanagement – система ведення господарства, що застосовується (зокрема інтенсивна, екстенсивна);

– barnphoto – фотографія цього приміщення господарства.

Таблиця Cattlevaccine містить дані про щеплення конкретних свійських тварин, включаючи стовпці:

– id – ідентифікатор проведеного щеплення свійської тварини;

– cattle – свійська тварина, щеплення якої було проведено, що визначається зовнішнім ключем з таблиці Cattle, формуючи відношення «один-до-багатьох»;

– vaccinetype – тип вакцини, яка застосовувалась при щепленні (проти сказу, ящура тощо);

– vaccinetime – час проведення вакцинації цього типу за цією твариною.

Таблиця Diseasetype містить можливі захворювання худоби, включаючи стовпці:

– id – ідентифікатор можливого захворювання худоби;

– diseasename – назва можливого захворювання худоби;

– diseasegroup – група захворювань худоби, до якої належить це захворювання.

Таблиця Cattlesale містить випадки, коли свійські тварини були продані або іншим чином переведені з одного господарство в інше, включаючи стовпці:

– id – ідентифікатор випадку переведення свійської тварини з одного господарства в інше;

– cattle – свійська тварина, яку було переведено з одного господарства в інше, що визначається зовнішнім ключем з таблиці Cattle, формуючи відношення «один-до-багатьох»;

– saledate – час, коли було продано або переведено цю свійську тварину з одного господарства до іншого;

– oldhousehold – господарство, якому до цієї події належала ця свійська тварина, що визначається зовнішнім ключем з таблиці Household, формуючи відношення «один-до-багатьох»;

– newhousehold – господарство, якому після цієї події належить ця свійська тварина, що визначається зовнішнім ключем з таблиці Household, формуючи відношення «один-до-багатьох»;

– salestate – стан продажу цієї свійської тварини (тільки поданий або підтверджений).

І значення в стовпці oldhousehold, і значення в стовпці newhousehold може бути не заповнено. Якщо перше не заповнено, то раніше ця свійська тварина належала господарству, яке не представлено в програмі. Якщо друге не заповнено, то тепер ця свійська тварина належить господарству, яке не представлено в програмі.

Таблиця Cattledisease містить випадки виявлення захворювань у свійських тварин, включаючи стовпці:

– id – ідентифікатор випадку виявлення захворювання у свійської тварини;

– cattle – свійська тварина, у якої виявлено цю хворобу, що визначається зовнішнім ключем з таблиці Cattle, формуючи відношення «один-до-багатьох»;

– diseasestart – час, коли хворобу було зафіксовано у цієї свійської тварини;

– diseaseend – час, з якого ця хвороба у цієї свійської тварини більше не спостерігається (якщо значення не задано, то хвороба ще не подолана);

– disease – хвороба, яка спостерігається у цієї свійської тварини, що визначається зовнішнім ключем з таблиці Diseasetype, формуючи відношення «один-до-багатьох» (рис. 3.2).

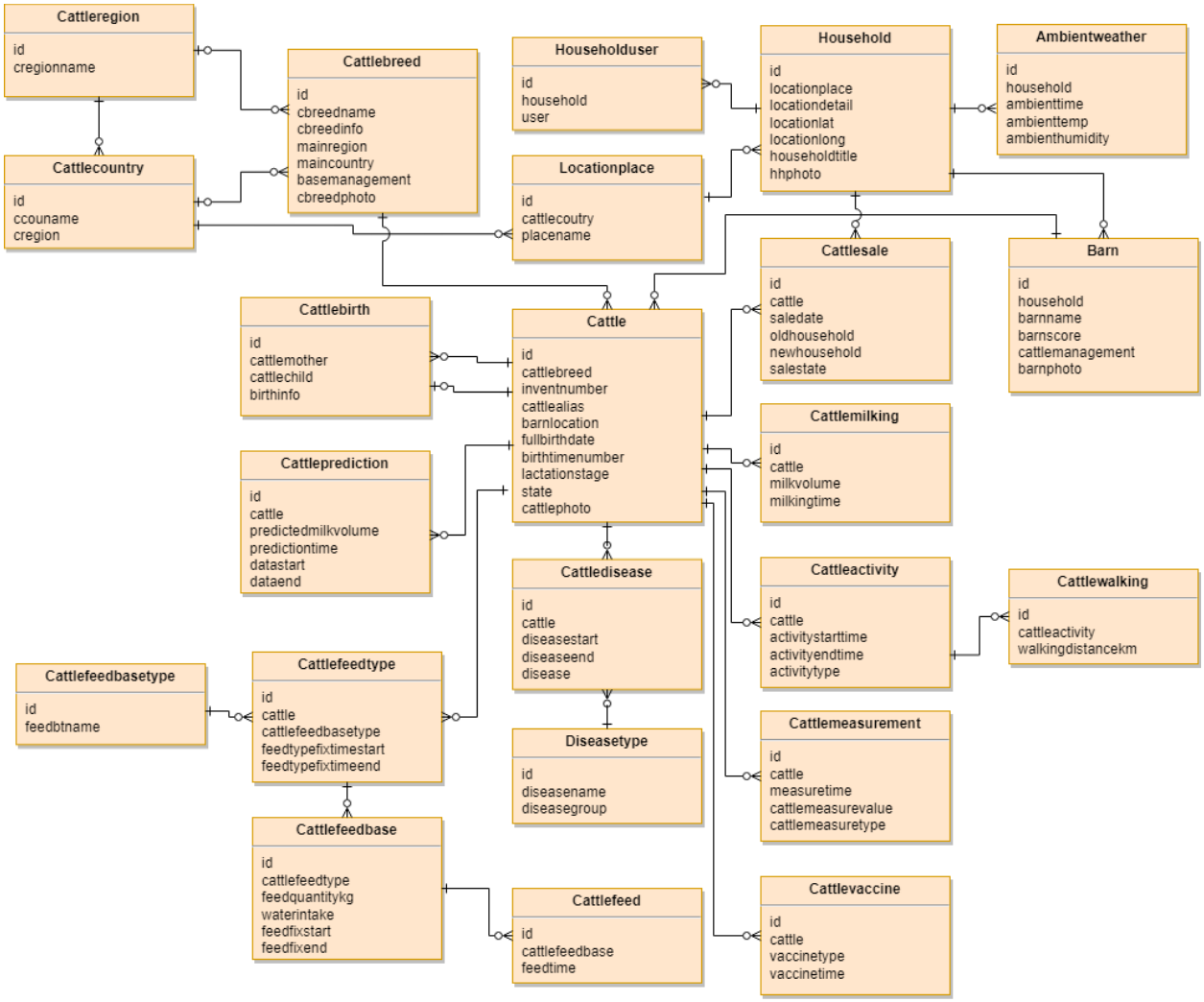

Рисунок 3.2 – Схема бази даних

3.4 Висновки за розділом 3

Сукупність засобів для розробки програми відслідковування стану худоби за результатами виконання третього розділу включає мову Python, фреймворк Django для забезпечення підтримки безпосередньо створення вебзастосунку, системи MySQL для організації бази даних. Моделювання сценаріїв забезпечено для користувача програми відслідковування стану худоби зі створенням діаграми прецедентів. Створено відповідну структуру бази даних, побудовано її схему.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Послідовність взаємодії користувача з програмою відслідковування стану худоби

Для роботи з програмою відслідковування стану худоби користувач обов’язково повинен авторизуватись (рис. 4.1).

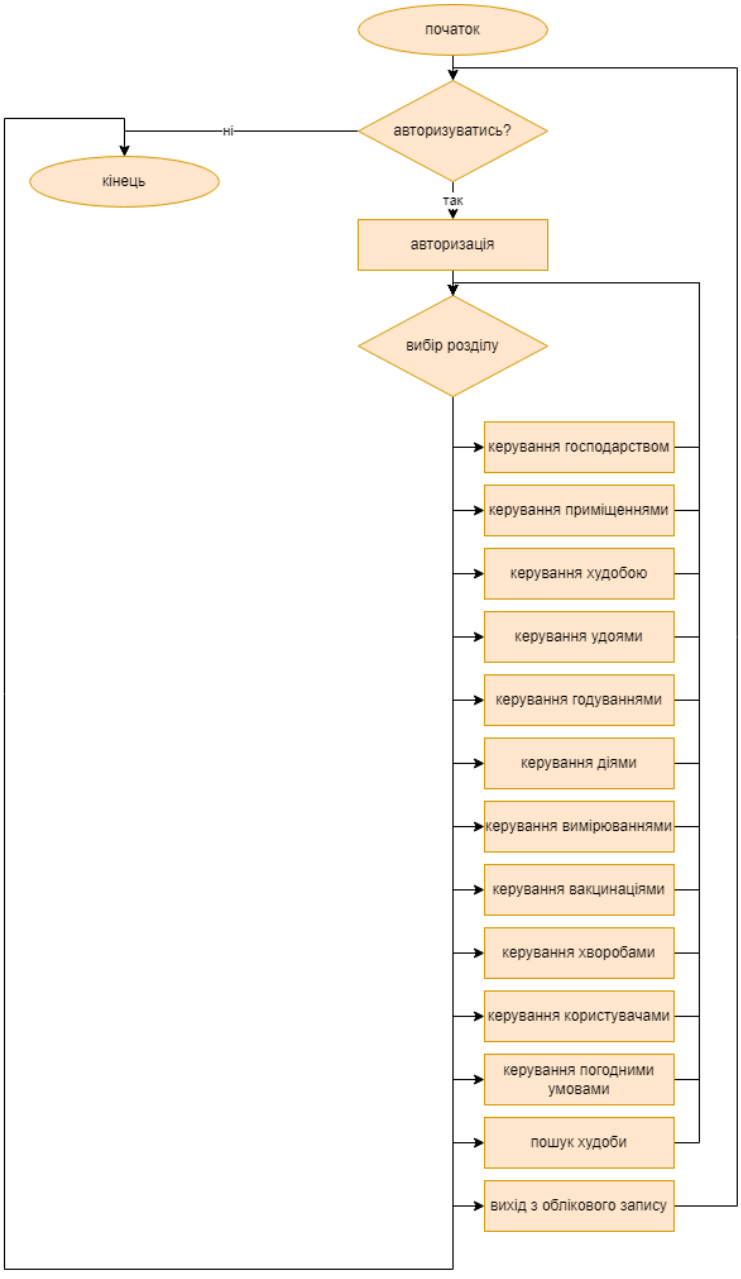

Рисунок 4.1 – Функціональна схема програми відслідковування стану худоби

Без авторизації користувач програми відслідковування стану худоби не може перейти до вибору розділів меню та почати працювати з даними. У такому випадку завершується робота з програмою. Якщо користувач авторизувався, то він може обрати потрібний йому з доступних розділів меню, враховуючи специфіку роботи з ними саме за тим господарством, до якого йому надано доступ, а також може вийти з облікового запису або просто завершити роботу з програмою.

4.2 Рішення з реалізації вебзастосунку

Рішення з реалізації охоплюють всі складові структури вебзастосунку відслідковування стану худоби, що включає базовий проєкт Django. Основний набір таких розроблених складових та прийнятих рішень описано нижче.

Представлення створені у вигляді функцій, забезпечуючи підтримку роботи зі сценаріями, описаними в розділі 3, для користувача та надаючи доступ до шаблонів вебсторінок, які створені під кожен сценарій на основі базового вільного шаблону [20].

Перегляд даних власного господарства реалізовано представленням як функцію household_view. Функція окрім стандартного параметра request більше не має параметрів, оскільки номер самого господарства береться з того, до якого саме господарства прив’язаний користувач, що авторизувався. Якщо він не авторизувався, дані не буде отримано. Всі дані представлені ключами в контексті даних, яким заповнюється шаблон вебсторінки household.html. Так за ключем household міститься сам об’єкт господарства, cattles – 5 тварин цього господарства, а їх загальна кількість – за ключем cattles_count, за ключем barns – 4 приміщення цього господарства, barns_count – загальна кількість таких приміщень цього господарства, breeds – 5 порід худоби цього господарства, breeds_count – загальна кількість таких порід, measurements – 5 останніх вимірювань стану худоби господарства, measurements_count – загальна кількість таких вимірювань, measurements_lastdate – остання дата

вимірювання, disease_active_count – кількість активних випадків хвороб худоби господарства, diseases – останні 5 таких випадків, vaccines – кількість діючих вакцинувань хвороби за вакцинами, vaccines_lastdate – дата останнього вакцинування, weathers – 5 останніх записів про погодні умови, weather_lastdate – дата останнього запису про погодні умови, milkings – 5 останніх записів про удої, milking_lastdate – дата останнього внесеного удою, milkings_30days_literes – сумарний обсяг молока в літрах, який було отримано від корів господарства за останні 30 днів, activities – останні 5 дій над худобою господарства, activity_lastdate – дата останньої дії над худобою господарства, activity_todo_number – кількість незавершених дій, feeds – 5 останніх записів про годування худоби господарства, feed_lastdate – дата останнього годування.

Перегляд користувачів, які представляють господарство, реалізовано представленням як функцію household_users_view.

Надання права працювати від імені господарства користувачу реалізовано представленням як функцію household_add_user_view, параметр якої username визначає логін користувача, якому ці права надаються.

Додавання приміщення до господарства реалізовано представленням як функцію household_add_barn_view, а внесення даних господарства забезпечується класом відповідної форми.

Перегляд всіх приміщень господарства реалізовано представленням як функцію household_barns_view, що має параметр page, який задає номер сторінки відображення таких приміщень (по 10 на сторінці).

Перегляд всієї худоби, зареєстрованої за господарством, реалізовано представленням як функцію household_cattles_view з параметрами page, що задає номер сторінки відображення худоби (по 10 на сторінці), barn_id, що задає ідентифікатор приміщення, cattle_breed_id, що задає іденифікатор породи худоби.

Реєстрація нової худоби господарством реалізовано представленням як функцію household_add_cattle_view, внесення даних худоби забезпечується класом відповідної форми.

Реєстрація продажу худоби реалізовано представленням як функцію cattle_sale_view з параметрами cattle_id, що задає ідентифікатор свійської тварини, household_id, що задає ідентифікатор господарства, якому продано цю свійську тварину.

Перегляд даних приміщення господарства реалізовано представленням як функцію household_barn_view з параметром bard_in, що задає ідентифікатор приміщення господарства.

Реєстрація придбання існуючої в програмі худоби реалізовано представленням як функцію cattle_buy_view з параметрами cattle_id, що задає ідентифікатор свійської тварини.

Перегляд всієї придбаної господарством худоби реалізовано представленням як функцію bought_cattles_view.

Перегляд всієї проданої господарством худоби реалізовано представленням як функцію sold_cattles_view.

Перегляд поголів’я народженої худоби в господарстві реалізовано представленням як функцію birthed_cattles_view.

Перегляд поголів’я народженої твариною худоби реалізовано представленням як функцію cattle_births_view з параметрами cattle_id, що задає ідентифікатор свійської тварини.

Перегляд динаміки народження худоби за господарством реалізовано представленням як функцію household_birth_dynamics_view.

Внесення удою худоби за твариною реалізовано представленням як функцію cattlemilking_add_view з параметром milkingtime, що є рядком, який задає час удою, volume, що задає обсяг молока.

Внесення удою худоби за породою тварин реалізовано представленням як функцію breed_cattlemilking_add_view, що працює на основі POST-запиту з параметром breed_id, що задає ідентифікатор породи худоби.

Внесення удою худоби за приміщенням господарства реалізовано представленням як функцію barn_cattlemilking_add_view, що працює на основі POST-запиту з параметром barn_id, що задає ідентифікатор приміщення.

Перегляд динаміки удоїв худоби загалом за господарством реалізовано представленням як функцію household_milking_dynamics з параметрами startdate та enddate, що у вигляді рядків задають дати проміжку, за якими формується помісячна динаміка (кількість сумарно зібраних літрів за кожен місяць періоду).

Перегляд динаміки удоїв худоби заданої породи за господарством реалізовано представленням як функцію breed_milking_dynamics з параметрами startdate та enddate, що у вигляді рядків задають дати проміжку, за якими формується помісячна динаміка (кількість сумарно зібраних літрів за кожен місяць періоду), breed_id, що задає ідентифікатор породи худоби.

Перегляд динаміки удоїв конкретної тварини реалізовано представленням як функцію cattle_milking_dynamics з параметрами startdate та enddate, що у вигляді рядків задають дати проміжку, за якими формується помісячна динаміка (кількість сумарно зібраних літрів за кожен місяць періоду), cattle_id, що задає ідентифікатор худоби.

Перегляд історії удоїв худоби за приміщенням господарства реалізовано представленням як функцію barn_milking_history_view з параметром barn_id, що задає ідентифікатор приміщення.

Перегляд прогнозувань удою худоби реалізовано представленням як функцію cattle_predictions_view з параметром page, що задає номер сторінки відображення..

Прогнозування удою худоби реалізовано представленням як функцію cattle_predict_view з параметром cattle_id, що задає ідентифікатор худоби. Безпосередньо прогнозування забезпечується на основі моделі дерева рішень, що побудована класом DecisionTreeRegressor з бібліотеки scikit-learn [21]-[22].

Встановлення базового варіанта годування худоби реалізовано представленням як функцію cattlefeedtype_add_view з параметром cattle_id, що задає ідентифікатор худоби, а внесення даних параметрів годування забезпечується класом відповідної форми..

Встановлення параметрів годування худоби реалізовано представленням як функцію cattlefeedbase_add_view з параметром cattlefeed_id, що задає ідентифікатор базового варіанта годування худоби, а внесення даних параметрів годування забезпечується класом відповідної форми.

Внесення годування худоби за твариною реалізовано представленням як функцію cattlefeed_add_view з параметром feedtime, що задає час годування, cattlefeedbase_id, що задає ідентифікатор параметрів годування худоби.

Внесення годування худоби за породою тварин реалізовано представленням як функцію breed_cattlefeed_add_view, що працює на основі POST-запиту з параметром breed_id, що задає ідентифікатор породи худоби.

Внесення годування худоби за приміщенням господарства реалізовано представленням як функцію barn_cattlefeed_add_view, що працює на основі POST-запиту з параметром barn_id, що задає ідентифікатор приміщення.

Перегляд історії годування худоби за приміщенням господарства реалізовано представленням як функцію barn_feeds_view з параметром barn_id, що задає ідентифікатор приміщення.

Перегляд історії годування худоби за твариною реалізовано представленням як функцію cattle_feeds_view з параметром cattle_id, що задає ідентифікатор свійської тварини, page, що задає номер сторінки відображення..

Перегляд історії зміни варіантів годування худоби за твариною реалізовано представленням як функцію cattle_feed_types_view з параметром cattle_id, що задає ідентифікатор свійської тварини.

Перегляд історії зміни параметрів годування худоби за твариною реалізовано представленням як функцію cattle_feed_bases_view з параметром cattle_id, що задає ідентифікатор свійської тварини.

Перегляд історії зміни варіантів годування худоби в господарстві за породою реалізовано представленням як функцію breed_feed_types_view з параметром breed_id, що задає ідентифікатор породи худоби.

Перегляд історії зміни параметрів годування худоби в господарстві за породою реалізовано представленням як функцію breed_feed_bases_view з параметром breed_id, що задає ідентифікатор породи худоби.

Перегляд історії зміни обсягів годування худоби в господарстві за варіантом годування реалізовано представленням як функцію feed_volumes_view з параметром cattlefeedbasetype_id, що задає ідентифікатор варіанта годування худоби.

Внесення дій над худобою за твариною реалізовано представленням як функцію cattle_activity_add_view з параметром cattle_id, що задає ідентифікатор свійської тварини, а внесення даних дій над худобою забезпечується класом відповідної форми.

Внесення дій над худобою за породою тварин реалізовано представленням як функцію breed_cattle_activity_add_view, що працює на основі POST-запиту з параметром breed_id, що задає ідентифікатор породи свійської тварини.

Внесення дій над худобою за приміщенням господарства реалізовано представленням як функцію barn_cattle_activity_add_view, що працює на основі POST-запиту з параметром barn_id, що задає ідентифікатор приміщення.

Визначення завершення дії за худобою реалізовано представленням як функцію cattle_activity_finish_view з параметром cattleactivity_id, що задає ідентифікатор дії, activitytime, що задає у вигляді рядка час завершення дії.

Перегляд дій над худобою за твариною реалізовано представленням як функцію cattle_activities_view з параметром cattle_id, що задає ідентифікатор свійської тварини, page, що задає номер сторінки відображення..

Перегляд дій над худобою за приміщенням господарства реалізовано представленням як функцію barn_cattle_activities_view з параметром barn_id, що задає ідентифікатор приміщення, page, що задає номер сторінки відображення..

Перегляд активності дій над худобою за господарством реалізовано представленням як функцію activities_tofinish_view.

Внесення вимірювань стану худоби за твариною реалізовано представленням як функцію cattlemeasurement_add_view з параметром cattle_id, що задає ідентифікатор свійської тварини, а внесення даних вимірювання стану худоби забезпечується класом відповідної форми.

Внесення вимірювань стану худоби за приміщенням господарства реалізовано представленням як функцію barn_cattle_measurements_add_view, що працює на основі POST-запиту з параметром barn_id, що задає ідентифікатор приміщення.

Внесення вимірювань стану худоби за породою реалізовано представленням як функцію breed_cattle_measurements_add_view, що працює на основі POST-запиту з параметром breed_id, що задає ідентифікатор породи свійської тварини.

Перегляд історії зміни стану худоби за твариною реалізовано представленням як функцію cattle_measurements_history_view з параметром cattle_id, що задає ідентифікатор свійської тварини.

Перегляд динаміки зміни стану худоби за приміщенням господарства реалізовано представленням як функцію barn_cattle_state_dynamics_view з параметрами startdate та enddate, що у вигляді рядків задають дати проміжку, за якими формується помісячна динаміка, barn_id, що задає ідентифікатор приміщення.

Перегляд динаміки зміни стану худоби за породою реалізовано представленням як функцію breed_cattle_state_dynamics_view з параметрами startdate та enddate, що у вигляді рядків задають дати проміжку, за якими формується помісячна динаміка, breed_id, що задає ідентифікатор породи худоби.

Внесення вакцинації худоби за твариною реалізовано представленням як функцію cattlevaccine_add_view з параметром cattle_id, що задає

ідентифікатор свійської тварини, а внесення даних вакцинації худоби забезпечується класом відповідної форми.

Внесення вакцинації худоби за приміщенням господарства реалізовано представленням як функцію barn_cattle_vaccinations_add_view, що працює на основі POST-запиту з параметром barn_id, що задає ідентифікатор приміщення.

Внесення вакцинації худоби за породою реалізовано представленням як функцію breed_cattle_vaccinations_add_view, що працює на основі POST-запиту з параметром breed_id, що задає ідентифікатор породи.

Перегляд історії вакцинації худоби за твариною реалізовано представленням як функцію cattle_vaccination_view з параметром cattle_id, що задає ідентифікатор свійської тварини, page, що задає номер сторінки відображення.

Перегляд стану вакцинації худоби за приміщенням господарства реалізовано представленням як функцію household_barn_vaccination_view з параметром barn_id, що задає ідентифікатор приміщення.

Перегляд стану вакцинації худоби за породою реалізовано представленням як функцію household_breed_vaccination_view з параметром cattle_breed_id, що задає ідентифікатор породи худоби.

Внесення хвороби худоби реалізовано представленням як функцію cattledisease_add_view з параметром cattle_id, що задає ідентифікатор свійської тварини, а внесення даних хвороби худоби забезпечується класом відповідної форми.

Визначення завершення хвороби худоби реалізовано представленням як функцію cattledisease_finish_view з параметром cattledisease_id, що задає ідентифікатор захворювання свійської тварини, finishtime, що у вигляді рядка задає дату завершення хвороби.

Перегляд історії захворювання худоби за твариною реалізовано представленням як функцію cattle_diseases_view з параметром cattle_id, що

задає ідентифікатор свійської тварини, page, що задає номер сторінки відображення.

Перегляд історії захворювання худоби за приміщенням господарства реалізовано представленням як функцію barn_diseases_view з параметром barn_id, що задає ідентифікатор приміщення, page, що задає номер сторінки відображення.

Перегляд історії захворювання худоби за породою реалізовано представленням як функцію breed_diseases_view з параметром cattle_breed_id, що задає ідентифікатор породи, page, що задає номер сторінки відображення.

Перегляд динаміки захворювання худоби за господарством реалізовано представленням як функцію household_disease_dynamics_view.

Перегляд динаміки захворювання худоби за приміщенням господарства реалізовано представленням як функцію barn_disease_dynamics_view з параметром barn_id, що задає ідентифікатор приміщення, startdate та enddate, що у вигляді рядків задають дати проміжку, за якими формується помісячна динаміка.

Перегляд динаміки захворювання худоби за породою реалізовано представленням як функцію breed_disease_dynamics_view з параметром cattle_breed_id, що задає ідентифікатор породи худоби, startdate та enddate, що у вигляді рядків задають дати проміжку, за якими формується помісячна динаміка.

Пошук худоби за власним господарством реалізовано представленням як функцію cattle_search_view, що працює на основі POST-запиту з параметром з параметром page, що задає номер сторінки відображення результатів.

Пошук худоби за іншим господарством реалізовано представленням як функцію host_cattle_search_view, що працює на основі POST-запиту з параметром з параметром page, що задає номер сторінки відображення результатів.

Підтвердження продажу худоби реалізовано представленням як функцію cattle_sale_approve_view з параметром cattlesale_id, що задає ідентифікатор продажу худоби.

Перегляд поданих заявок придбань худоби господарства реалізовано представленням як функцію cattle_buy_applications_view з параметром page, що задає номер сторінки відображення результатів.

Перегляд власних поданих заявок придбань худоби реалізовано представленням як функцію cattle_sale_applications_view з параметром page, що задає номер сторінки відображення результатів.

Внесення погодних умов за господарством реалізовано представленням як функцію ambientweather_add_view, а внесення даних хвороби худоби забезпечується класом відповідної форми.

Перегляд внесених погодних умов за господарством реалізовано представленням як функцію household_ambientweathers_view з параметрами startdate та enddate, що у вигляді рядків задають дати проміжку часу.

Скасування права працювати від імені господарства користувачу реалізовано представленням як функцію household_user_cancel_view з параметром username, що задає логін користувача.

Редагування даних господарства реалізовано представленням як функцію household_edit_view з параметром household_id, що задає ідентифікатор господарства, а зміна даних господарства забезпечується класом відповідної форми.

Редагування даних приміщення господарства реалізовано представленням як функцію barn_edit_view з параметром barn_id, що задає ідентифікатор приміщення господарства, а зміна даних господарства забезпечується класом відповідної форми.

Вилучення приміщення з господарства реалізовано представленням як функцію barn_delete_view з параметром barn_id, що задає ідентифікатор приміщення господарства.

Редагування даних зареєстрованої худоби реалізовано представленням як функцію cattle_edit_view з параметром cattle_id, що задає ідентифікатор свійської тварини, а зміна даних господарства забезпечується класом відповідної форми.

Зняття худоби з обліку реалізовано представленням як функцію cattle_remove_view з параметром cattle_id, що задає ідентифікатор свійської тварини.

Редагування внесеного удою худоби реалізовано представленням як функцію cattle_milking_edit_view з параметром cattlemilking_id, що задає ідентифікатор удою худоби, а зміна даних удою худоби забезпечується класом відповідної форми.

Вилучення внесеного удою худоби реалізовано представленням як функцію cattle_milking_delete_view з параметром cattlemilking_id, що задає ідентифікатор удою худоби.

Редагування внесеного годування худоби реалізовано представленням як функцію cattle_feed_edit_view з параметром cattlefeed_id, що задає ідентифікатор годування худоби, а зміна даних годування худоби забезпечується класом відповідної форми.

Вилучення внесеного годування худоби реалізовано представленням як функцію cattle_feed_delete_view з параметром cattlefeed_id, що задає ідентифікатор годування худоби.

Редагування внесеної дії за худобою реалізовано представленням як функцію cattle_activity_edit_view з параметром cattleactivity_id, що задає ідентифікатор дії за худобою, а зміна даних дії за худобою забезпечується класом відповідної форми.

Вилучення внесеної дії за худобою реалізовано представленням як функцію cattle_activity_delete_view з параметром cattleactivity_id, що задає ідентифікатор дії за худобою.

Вилучення внесеної хвороби тварини реалізовано представленням як функцію cattle_disease_delete_view з параметром cattledisease_id, що задає ідентифікатор внесеної хвороби тварини.

Вилучення внесеної вакцинації худоби реалізовано представленням як функцію cattle_vaccination_delete_view з параметром cattlevaccination_id, що задає ідентифікатор внесеної вакцинації худоби.

Скасування даних продажу худоби реалізовано представленням як функцію cattle_sale_cancel_view з параметром cattlesale_id, що задає ідентифікатор продажу худоби.

Вилучення даних продажу худоби реалізовано представленням як функцію cattle_sale_delete_view з параметром cattlesale_id, що задає ідентифікатор продажу худоби.

Редагування внесених погодних умов за господарством реалізовано представленням як функцію ambientweather_edit_view з параметром ambientweather_id, що задає ідентифікатор внесених погодних умов за господарством, а зміна даних внесених погодних умов за господарством забезпечується класом відповідної форми.

Редагування внесеного вимірювання стану худоби реалізовано представленням як функцію cattle_measurement_edit_view з параметром cattlemeasurement_id, що задає ідентифікатор внесеного вимірювання стану худоби, а зміна даних внесеного вимірювання стану худоби забезпечується класом відповідної форми.

Вилучення внесеного вимірювання стану худоби реалізовано представленням як функцію cattle_measurement_delete_view з параметром cattlemeasurement_id, що задає ідентифікатор внесеного вимірювання стану худоби.

Вилучення невикористаного базового варіанта годування худоби реалізовано представленням як функцію cattle_feedtype_delete_view з параметром cattlefeedtype_id, що задає ідентифікатор базового варіанта годування худоби.

Редагування невикористаного базового варіанта годування худоби реалізовано представленням як функцію cattle_feedtype_edit_view з параметром cattlefeedtype_id, що задає ідентифікатор базового варіанта годування худоби, а зміна даних внесеного базового варіанта годування худоби забезпечується класом відповідної форми.

Редагування невикористаних параметрів годування худоби реалізовано представленням як функцію cattle_feedbase_edit_view з параметром cattlefeedbase_id, що задає ідентифікатор параметрів годування худоби, а зміна даних внесених параметрів годування худоби забезпечується класом відповідної форми.

Вилучення невикористаних параметрів годування худоби реалізовано представленням як функцію cattle_feedbase_delete_view з параметром cattlefeedbase_id, що задає ідентифікатор параметрів годування худоби.

Модель Cattleregion є класом, що визначає світові регіони розповсюдження худоби. Клас Cattleregion має поле cregionname класу models.CharField з назвою світового регіону розповсюдження худоби.

Модель Cattlecountry є класом, що визначає країни, в яких може вирощуватись худоба, в яких можуть знаходитись господарства.

Поле ccouname має клас models.CharField з назвою країни.

Поле cregion має клас models.ForeignKey на основі моделі Cattleregion, що задає світовий регіон розповсюдження худоби, де знаходиться ця країна.

Модель Cattlebreed є класом, що визначає породи худоби.

Поле cbreedname має клас models.CharField з назвою породи худоби.

Поле cbreedinfo має клас models.TextField з текстовою інформацією стосовно особливостей вирощування цієї породи худоби.

Поле mainregion має клас models.ForeignKey на основі моделі Cattleregion, що задає основний світовий регіон, де розповсюджена худоба цієї породи.

Поле maincountry має клас models.ForeignKey на основі моделі Cattlecountry, що задає основну країну, де вирощується худоба цієї породи. Це поле може мати порожнє значення.

Поле basemanagement має клас models.CharField з системою ведення господарства, яка зазвичай має застосовуватися для такої породи худоби. Цей клас визначений на основі словника FARMING_TYPES, де кодування таких систем ведення господарства задано.

Поле cbreedphoto має клас models.ImageField з фотографією худоби цієї породи.

Модель Cattle є класом, що визначає конкретних тварин.

Поле cattlebreed має клас models.ForeignKey на основі моделі Cattlebreed, що задає породу худоби, до якої ця свійська тварина належить.

Поле inventnumber має клас models.CharField з інвентарним номером цієї свійської тварини серед всієї худоби господарства. Це поле може мати порожнє значення.

Поле cattlealias має клас models.CharField з кличкою цієї свійської тварини. Це поле може мати порожнє значення.

Поле barnlocation має клас models.ForeignKey на основі моделі Barn, що задає приміщення господарства, в якому знаходиться ця свійська тварина.

Поле fullbirthdate має клас models.DateField з датою народження цієї свійської тварини.

Поле birthtimenumber має клас models.PositiveSmallIntegerField з кількістю худоби, яку народила ця свійська тварина (за час спостережень за нею).

Поле lactationstage має клас models.CharField з фазою лактації, в якій на даний момент знаходиться ця свійська тварина. Цей клас визначений на основі словника LACTATION_STAGES, де кодування таких фаз задано:

– роздоювання – символом “E”;

– стійкий удій – символом “M”;

– зниження – символом “L”;

– сухостійний період – символом “F”.

Поле state має клас models.CharField зі станом, в якому на даний момент знаходиться ця свійська тварина. Цей клас визначений на основі словника STATES, де кодування таких станів задано:

– активна – символом “A”;

– хвороба – символом “D”;

– неактивна – символом “I”.

Поле cattlephoto має клас models.ImageField з фотографією цієї свійської тварини.

Модель Cattlebirth є класом, що визначає народження худоби в господарстві.

Поле cattlemother має клас models.ForeignKey на основі моделі Cattle, що задає свійську тварину, яка народила. Зворотне звернення за цим зв’язком визначено за ім’ям children, тобто з моделі Cattle можна отримати доступ до колекції народженої худоби цією свійською твариною через це ім’я.

Поле cattlechild має клас models.OneToOneField на основі моделі Cattle, що задає свійську тварину, яку було народжено. Зворотне звернення за цим зв’язком визначено за ім’ям birth, тобто з моделі Cattle можна отримати доступ до даних народження цієї свійської тварини через це ім’я.

Поле birthinfo має клас models.CharField з додатковою інформацією стосовно народження цієї худоби. Це поле може мати порожнє значення.

Модель Cattlemilking є класом, що визначає дані молочної продуктивності худоби.

Поле cattle має клас models.ForeignKey на основі моделі Cattle, що задає свійську тварину, удій молока якої було проведено.

Поле milkvolume має клас models.DecimalField з обсягом молока, який було отримано під час цього удою цієї свійської тварини.

Поле milkingtime має клас models.DateTimeField з часом початку цього удою цієї свійської тварини.

Модель Cattleprediction є класом, що визначає прогнозування удоїв молока худоби.

Поле cattle має клас models.ForeignKey на основі моделі Cattle, що задає свійську тварину, прогнозування удою молока якої було виконано.

Поле predictedmilkvolume має клас models.DecimalField з обсягом удою молока, який буде в середньому отримано за наступний період (тиждень) для цієї свійської тварини за результатами прогнозування.

Поле predictiontime має клас models.DateTimeField з часом виконання прогнозування удою молока цієї худоби.

Поле datastart має клас models.DateTimeField з часом, починаючи з якого дані про стан, удій цієї свійської тварини були використані при прогнозуванні для усереднення в якості вхідних даних моделі.

Поле dataend має клас models.DateTimeField з часом, до якого дані про стан, удій цієї свійської тварини були використані при прогнозуванні для усереднення в якості вхідних даних моделі.

Модель Cattlefeedbasetype є класом, що визначає типові варіанти харчування худоби.

Поле feedbtname має клас models.CharField з назвою типового варіанта харчування худоби.

Модель Cattlefeedtype є класом, що визначає варіанти харчування, які застосовуються до конкретних свійських тварин.

Поле cattle має клас models.ForeignKey на основі моделі Cattle, що задає свійську тварину, до якої застосовується цей варіант харчування.

Поле cattlefeedbasetype має клас models.ForeignKey на основі моделі Cattlefeedbasetype, що задає типовий варіант харчування, який застосовується для цієї свійської тварини у цей період часу.

Поле feedtypefixtimestart має клас models.DateTimeField з часом, з якого почалось застосування цього варіанта харчування для цієї свійської тварини.

Поле feedtypefixtimeend має клас models.DateTimeField з часом, до якого тривало застосування цього варіанта харчування для цієї свійської

тварини (якщо не задано значення, то такий варіант харчування є поточним). Це поле може мати порожнє значення.

Модель Cattlefeedbase є класом, що визначає деталізацію варіантів харчування, які застосовуються до конкретних свійських тварин.

Поле cattlefeedtype має клас models.ForeignKey на основі моделі Cattlefeedtype, що задає варіант харчування, який застосовується до цієї свійської тварини та який деталізовано таким чином.

Поле feedquantitykg має клас models.DecimalField з обсягом харчування за цим варіантом, що застосовується для цієї свійської тварини, заданий в кілограмах.

Поле waterintake має клас models.DecimalField з обсягом води, який використовується для пиття цієї свійської тварини в літрах за час між харчуваннями.

Поле feedfixstart має клас models.DateTimeField з часом, з якого діє цей обсяг (деталізація) варіанта харчування цієї свійської тварини.

Поле feedfixend має клас models.DateTimeField з часом, до якого діє цей обсяг (деталізація) варіанта харчування цієї свійської тварини (якщо не задано значення, то така деталізація цього варіанта харчування є поточною). Це поле може мати порожнє значення.

Модель Cattlefeed є класом, що визначає випадки годування свійських тварин.

Поле cattlefeedbase має клас models.ForeignKey на основі моделі Cattlefeedbase, що задає деталізацію варіанта харчування свійської тварини, яку було застосовано під час цього годування.

Поле feedtime має клас models.DateTimeField з часом початку цього годування цієї свійської тварини.

Модель Cattleactivity є класом, що визначає дії (відпочинок, випас, перехід на пасовисько), які виконувались стосовно свійських тварин.

Поле cattle має клас models.ForeignKey на основі моделі Cattle, що задає свійську тварину, до якої було виконано цю дію.

Поле activitystarttime має клас models.DateTimeField з часом початку реалізації цієї дії до цієї свійської тварини.

Поле activityendtime має клас models.DateTimeField з часом завершення реалізації цієї дії до цієї свійської тварини. Це поле може мати порожнє значення.

Поле activitytype має клас models.CharField з типом дії, яку було виконано до цієї свійської тварини. Цей клас визначений на основі словника ACTIVITY_TYPES, де кодування таких типів дій задано:

– відпочинок – символом “R”;

– випас – символом “G”;

– перехід – символом “W”.

Модель Cattlewalking є класом, що визначає дані про перехід тварин на пасовисько.

Поле cattleactivity має клас models.OneToOneField на основі моделі Cattleactivity, що задає дію, якою було описано всю стандартну частину цього переходу тварини на пасовисько.

Поле walkingdistancekm має клас models.DecimalField з відстанню в кілометрах, яку подолала ця свійська тварина під час такого переходу за цією дією.

Модель Cattlemeasurement є класом, що визначає вимірювання стану (температури, серцебиття, частоти дихання, стану тіла) свійських тварин.

Поле cattle має клас models.ForeignKey на основі моделі Cattle, що задає свійську тварину, вимірювання стану якої було виконано.

Поле measuretime має клас models.DateTimeField з часом, коли це вимірювання стану цієї свійської тварини було виконано.

Поле cattlemeasurevalue має клас models.DecimalField зі значенням показника, яке було виміряно для цієї світської тварини у цю мить.

Поле cattlemeasuretype має клас models.CharField з типом показника, який було виміряно для цієї світської тварини у цю мить. Цей клас визначений

на основі словника MEASURE_TYPES, де кодування таких типів показника задано:

– температура – символом “T”;

– серцебиття – символом “H”;

– дихання – символом “B”;

– загальний стан – символом “S”.

Модель Ambientweather є класом, що визначає спостереження за погодними умовами навколишнього середовища.

Поле household має клас models.ForeignKey на основі моделі Household, що задає господарство, де таке спостереження відбувалось.

Поле ambienttime має клас models.DateTimeField з часом спостереження за погодними умовами навколишнього середовища.

Поле ambienttemp має клас models.DecimalField з температурою навколишнього середовища в цей час в цьому населеному пункті.

Поле ambienthumidity має клас models.DecimalField з вологістю повітря навколишнього середовища в цей час в цьому населеному пункті.

Модель Locationplace є класом, що визначає місця (населені пункти), де можуть знаходитись господарства, відбуватися спостереження за погодними умовами навколишнього середовища.

Поле cattlecoutry має клас models.ForeignKey на основі моделі Cattlecountry, що задає країну, де це місце (населений пункт) знаходиться.

Поле placename має клас models.CharField з назвою місця (населеного пункту).

Модель Household є класом, що визначає господарства, які використовуючи програму, володіють худобою.

Поле locationplace має клас models.ForeignKey на основі моделі Locationplace, що задає місце (населений пункт), де знаходиться це господарство.

Поле locationdetail має клас models.CharField з деталізацією місця розташування господарства в цьому населеному пункті або біля нього у вигляді текстового опису (можливо конкретна адреса).

Поле locationlat має клас models.DecimalField з широтою місця розташування господарства (для відображення на мапі).

Поле locationlong має клас models.DecimalField з довготою місця розташування господарства (для відображення на мапі).

Поле householdtitle має клас models.CharField з назвою господарства.

Поле users має клас models.ForeignKey на основі моделі AUTH_USER_MODEL, що задає облікові записи користувачів, які можуть працювати з даними цього господарства на основі відповідної базової Django-моделі для автентифікації користувачів.

Поле hhphoto має клас models.ImageField з фотографією господарства.

Модель Barn є класом, що визначає приміщення господарства, в яких знаходиться худоба.

Поле household має клас models.ForeignKey на основі моделі Household, що задає господарство, яке володіє цим приміщенням, де може знаходитись худоба.

Поле barnname має клас models.CharField з назвою приміщення, якщо воно має конкретну назву, або певна позначка для текстової ідентифікації.

Поле barnscore має клас models.IntegerField з оцінкою стану приміщення стосовно збереження в ньому худоби від 1 до 5.

Поле cattlemanagement має клас models.CharField з системою ведення господарства, що застосовується. Цей клас визначений на основі словника FARMING_TYPES.

Поле barnphoto має клас models.ImageField з фотографією цього приміщення господарства.

Модель Cattlevaccine є класом, що визначає дані про щеплення конкретних свійських тварин.

Поле cattle має клас models.ForeignKey на основі моделі Cattle, що задає свійську тварину, щеплення якої було проведено.

Поле vaccinetype має клас models.CharField з типом вакцини, яка застосовувалась при щепленні. Цей клас визначений на основі словника VACCINE_TYPES.

Поле vaccinetime має клас models.DateTimeField з часом проведення вакцинації цього типу за цією твариною.

Модель Diseasetype є класом, що визначає можливі захворювання худоби.

Поле diseasename має клас models.CharField з назвою можливого захворювання худоби.

Поле diseasegroup має клас models.CharField з групою захворювань худоби, до якої належить це захворювання. Цей клас визначений на основі словника DISEASE_GROUPS.

Модель Cattledisease є класом, що визначає випадки виявлення захворювань у свійських тварин.

Поле cattle має клас models.ForeignKey на основі моделі Cattle, що задає свійську тварину, у якої виявлено цю хворобу.

Поле diseasestart має клас models.DateTimeField з часом, коли хворобу було зафіксовано у цієї свійської тварини.

Поле diseaseend має клас models.DateTimeField з часом, з якого ця хвороба у цієї свійської тварини більше не спостерігається (якщо значення не задано, то хвороба ще не подолана). Це поле може мати порожнє значення.

Поле disease має клас models.ForeignKey на основі моделі Diseasetype, що задає хворобу, яка спостерігається у цієї свійської тварини.

Модель Cattlesale є класом, що визначає випадки, коли свійські тварини були продані або іншим чином переведені з одного господарство в інше.

Поле cattle має клас models.ForeignKey на основі моделі Cattle, що задає свійську тварину, яку було переведено з одного господарства в інше.

Поле saledate має клас models.DateField з часом, коли було продано або переведено цю свійську тварину з одного господарства до іншого.

Поле oldhousehold має клас models.ForeignKey на основі моделі Household, що задає господарство, якому до цієї події належала ця свійська тварина. Зворотне звернення за цим зв’язком визначено за ім’ям saledcattles, тобто з моделі Household можна отримати доступ до колекції проданої худоби через це ім’я.

Поле newhousehold має клас models.ForeignKey на основі моделі Household, що задає господарство, якому після цієї події належить ця свійська тварина. Зворотне звернення за цим зв’язком визначено за ім’ям boughtcattles, тобто з моделі Household можна отримати доступ до колекції купленої худоби через це ім’я.

Поле salestate має клас models.BooleanField зі станом продажу цієї свійської тварини (тільки поданий або підтверджений).

4.3 Висновки за розділом 4

Вебзастосунок відслідковування стану худоби повністю програмно реалізовано, а рішення, які були прийняті та стали основою створених програмних елементів, описано в розділі.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Опис програми

Програма відслідковування стану худоби необхідна користувачам для керування даними худоби за тим господарством, яке вони представляють, при цьому контролюючи цілу сукупність станів, дій, які пов’язані зі щоденною роботою з худобою. До цього входять загальні дані, дані вакцинацій, хвороб, фактичних і прогнозованих зборів молока, переходів на пасовиська, вимірювання стану свійської тварини, народження, продажу або придбання тощо.

5.2 Експлуатація програми

Стандартним початком роботи з програмою відслідковування стану худоби для будь-якого користувача є заповнення даних у формі авторизації, а за необхідності виконання переходу до форми реєстрації. Після заповнення форми реєстрації користувачу має бути надано доступ до даних господарства одним з користувачів, які такі права для роботи з господарством мають. Якщо права підтверджені, то тоді за всіма розділами меню користувачу буде надано можливість працювати з відповідними даними за його господарством. Це буде відбуватися після його авторизації.

Початковою сторінкою для роботи є сторінка керування даними господарства (рис. 5.1).

На сторінці господарства користувачу доступні загальні дані господарства, а мапа з точкою його розташування розташована в самому низу сторінки. Доступна кнопка для переходу до редагування даних за цим господарством.

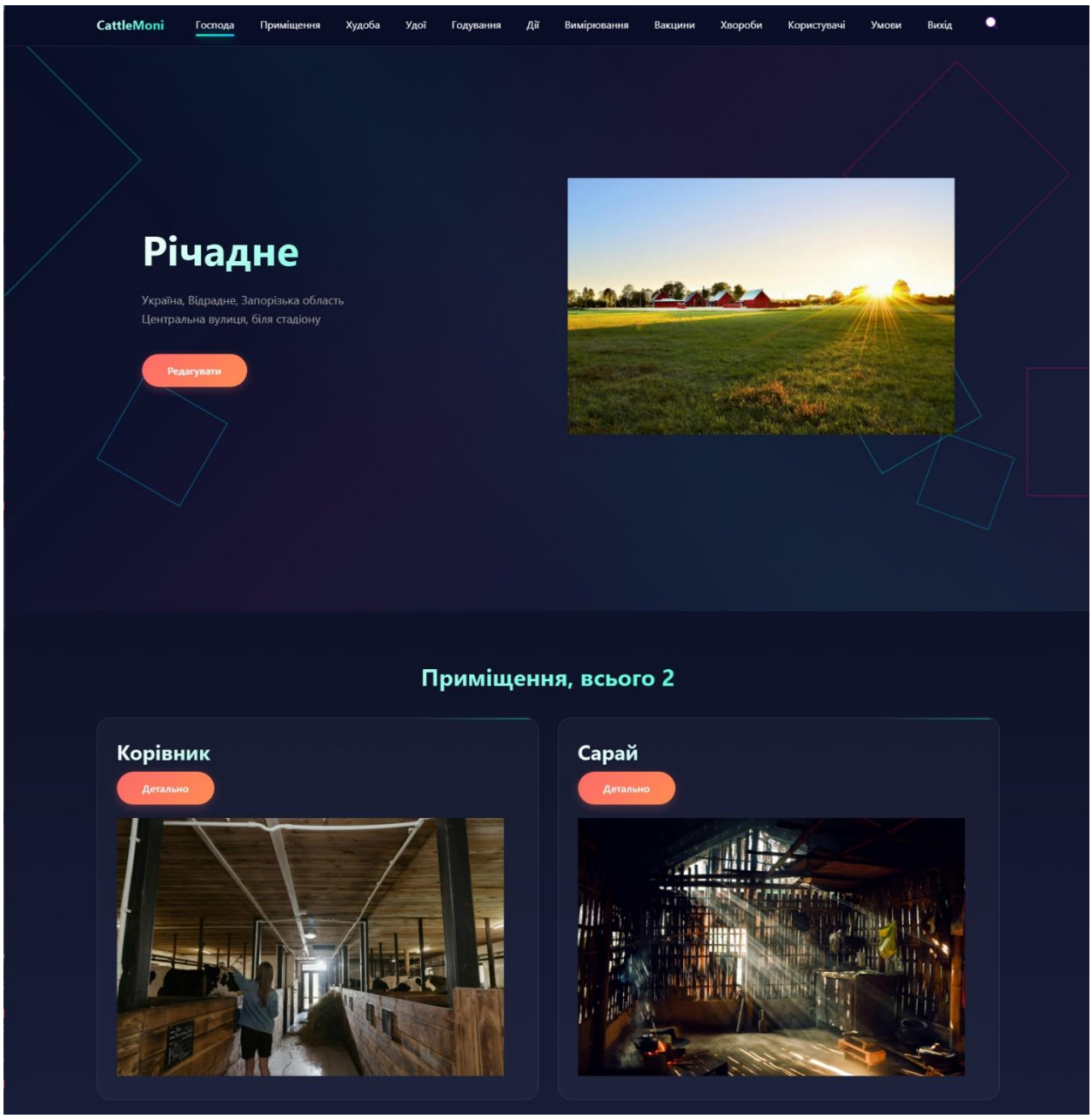

Рисунок 5.1 – Перегляд загальних даних господарства на його сторінці

На сторінці також розташовано дані 5 тварин цього господарства з посиланням до переходу на сторінку даних про всі тварини, якщо їх сумарно більше 5. Також доступні дані 4 приміщень цього господарства (рис. 5.1) та посилання на сторінку зі всіма, якщо їх більше. Окрім того доступні дані 5 порід худоби цього господарства та посилання на всі, якщо їх більше, дані 5 останніх вимірювань стану худоби господарства та посилання на сторінку зі всіма, остання дата вимірювання, кількість активних випадків хвороб худоби господарства, останні 5 таких випадків, посилання на всі, кількість діючих

вакцинувань хвороби за вакцинами (рис. 5.2), дата останнього вакцинування, дані 5 останніх записів про погодні умови та посилання на всі, дата останнього запису про погодні умови, 5 останніх записів про удої та посилання на всі, дата останнього внесеного удою, сумарний обсяг молока в літрах, який було отримано від корів господарства за останні 30 днів, останні 5 дій над худобою господарства та посилання на всі, дата останньої дії над худобою господарства, кількість незавершених дій, 5 останніх записів про годування худоби господарства та посилання на всі, дата останнього годування, посилання на перехід до продажів та придбань тварин.

Рисунок 5.2 – Перегляд стану активних вакцинувань за тваринами господарства на його сторінці

Для того щоб самому визначити придбання худоби в іншого господарства, потрібно перейти на сторінку пошуку з натисканням на зображення лупи в меню у правому краї, тоді обрати простір пошуку по іншим

господарствам, задати параметри пошуку, а тоді за знайденою твариною оформити заявку стосовно придбання. Цю заявку далі зі сторінки продажу і придбання має підтвердити сторона, яка є власником цієї свійської тварини.

Розділ приміщень у меню дозволяє переглядати дані всіх наявних приміщень господарства, переходити до редагування їх даних, вилучення, перегляду даних стосовно худоби за приміщенням.

Розділ худоби дозволяє зареєструвати нову худобу, якщо її даних не було в програмі за іншим господарством раніше і вона не народжена в цьому господарстві (рис. 5.3).

Рисунок 5.3 – Худоба за породою в програмі

На сторінці також можна відобразити худобу за обраною породою (позначається окремим позиціями для вибору назви, якщо порід декілька за господарством) або перейти до худоби за приміщенням за допомогою відповідної кнопки. За кожною свійською твариною відображаються дані про неї, надається можливість зареєструвати народження нею тварини, продаж, виконати редагування даних, зняти тварину з обліку (зокрема якщо вона померла).

Розділ вакцинувань дозволяє обирати вакцинування за породою (рис. 5.4) або приміщенням, а тоді за кожною вакциною бачити наявність активного вакцинування та дату, до якої вона діє, переходити до вилучення таких даних.

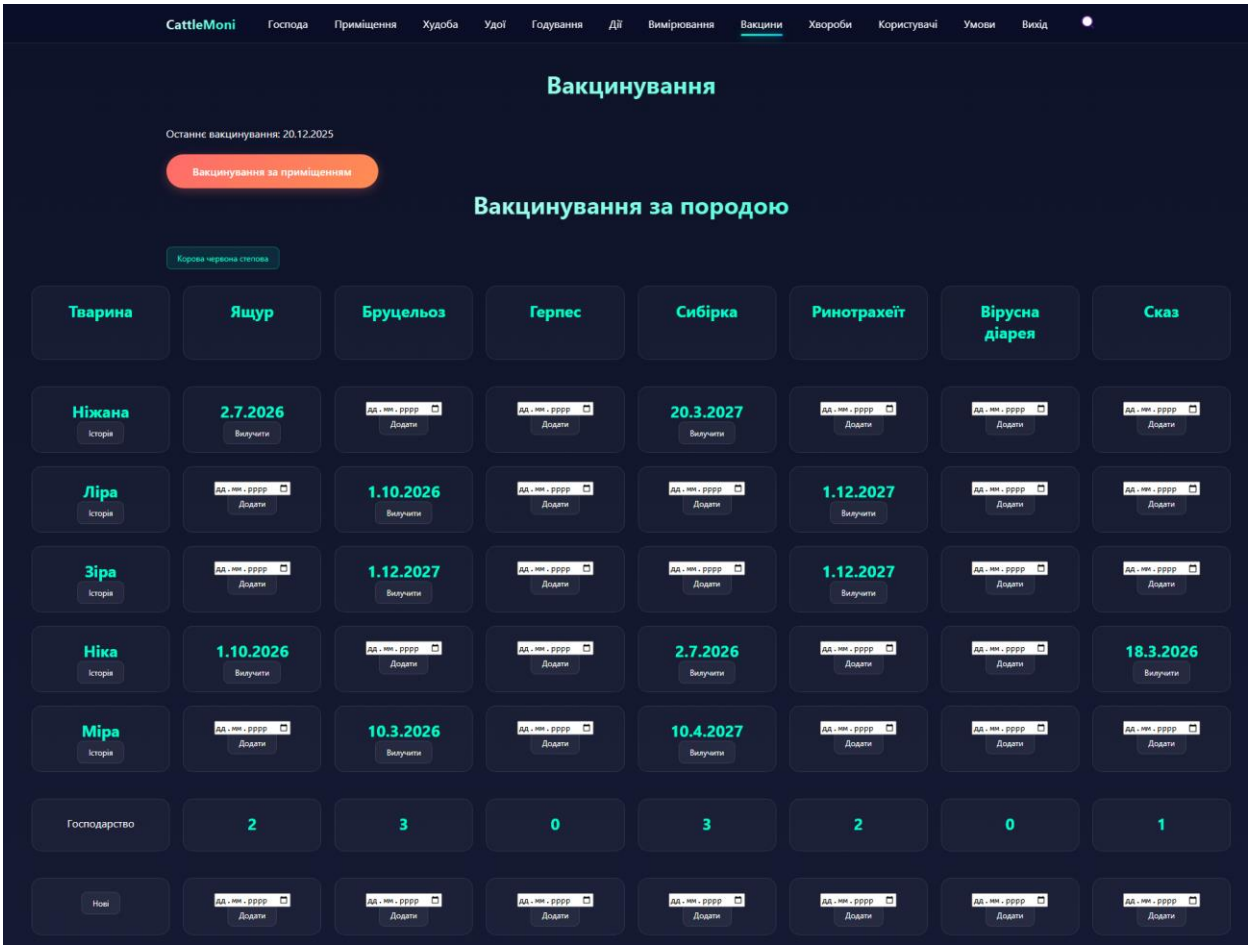

Рисунок 5.4 – Керування вакцинаціями худоби за породою

У випадку відсутності такого вакцинування доступна можливість ввести дату, до якої проведене вакцинування діє, та підтвердити ці дані. Внести відповідні дані можна також за всіма тваринами приміщення господарства. Для цього потрібно в нижньому рядку (передостанній рядок містить кількість активних вакцинувань сумарно) задати таку дату і натиснути на посилання додати. Тоді буде внесено вакцинування за кожною твариною за цією вакциною.

Розділ хвороб дозволяє переходити між історією в різних варіантів, де відображаються конкретні випадки захворювання, та динамікою (рис. 5.5).

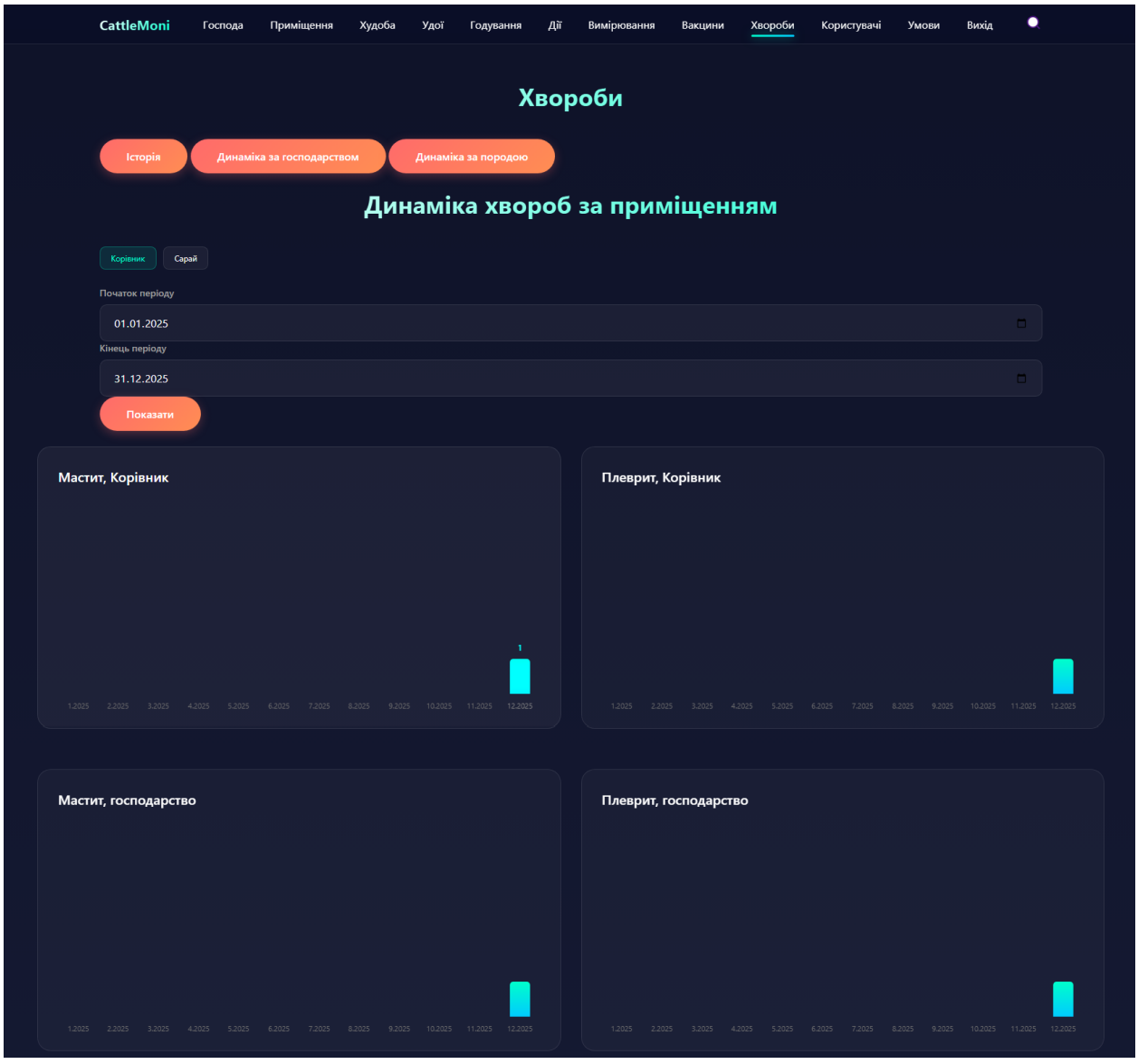

Рисунок 5.5 – Динаміка хвороб за приміщенням у програмі

У випадку, якщо відкритий варіант динаміки, то може бути обрано або динаміку за приміщенням (рис. 5.5), або за господарством, або за породою. Динаміка відображається за обраний період часу або за останні 12 місяців. Вибираються ті захворювання, які були виявлені, а за кожним показується місячна кількість на графіках за приміщенням та за господарством загалом.

Визначати доступ користувачів для роботи з господарством можна через розділ «Користувачі».

Розділ «Годування» дозволяє за допомогою відповідних посилань переходити до визначення базових варіантів годування для тварин, визначення конкретних параметрів, визначення випадків годування за цими параметрами за датами.

Розділ «Дії» дозволяє вносити, переглядати, змінювати, завершувати дії над худобою господарства. Для цього також доступні відповідні посилання у вигляді кнопок.

Розділ «Вимірювання» дозволяє фіксувати виконання відповідних дій над тваринами. Внесення вимірювань стану худоби відбувається подібно до вакцинації або за твариною, або за приміщенням господарства, або за породою, при цьому останні 2 варіанти дозволяють скоротити час, якщо результати є стандартними. Можна також переглядати історію того, як змінювались стани тварин за час спостережень.

Розділ «Удої» містить можливість і фіксувати виконані удої, переглядати статистику, але також і ініціювати на сторінці удоїв за худобою прогнозування. Результати прогнозувань відображаються за кожною твариною у загальній таблиці зі всіма тваринами господарства. У таблиці демонструється останнє прогнозоване значення для тварини, а також дата такого прогнозування (рис. 5.6).

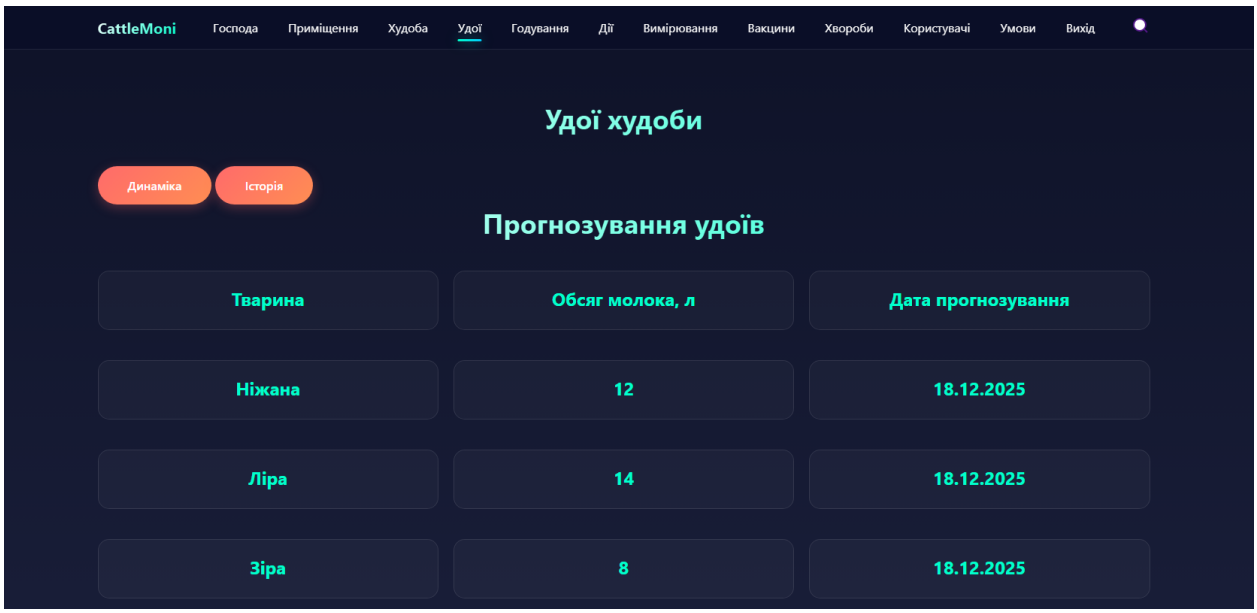

Рисунок 5.6 – Прогнозування результатів збору молока за худобою господарства

5.3 Експериментальне дослідження

Проведення експериментального дослідження виконано на основі вибірки даних [23], що містить 250 тисяч записів про худобу з наступними ознаками:

– номер свійської тварини;

– кількість народженої худоби свійською твариною;

– порода худоби;

– кількість послідовних днів, протягом якої дає молоко ця свійська тварина;

– кількість води, яку всередньому споживає на день ця тварина;

– середня температура повітря на прогнозований період часу;

– відсоток вологості повітря на прогнозований період часу;

– вага їжі, яку споживає на день всередньому свійська тварина;

– сезон, на який відбувається прогнозування;

– середня кількість кілометрів, яку долає на день свійська тварина;

– поточна повна кількість місяців віку свійської тварини;

– регіон розміщення породи худоби;

– оцінка приміщення, в якому знаходиться худоба (від 1 до 5);

– середня кількість годин відпочинку свійської тварини на добу;

– середня кількість годин випасання свійської тварини на добу;

– країна знаходження свійської тварини;

– тип годування свійської тварини;

– клімат знаходження свійської тварини;

– середня кількість годин проміжку між удоями;

– оцінка стану тіла худоби;

– частота годування свійської тварини на добу;

– система ведення господарства;

– стадія лактації свійської тварини;

– середня кількість в літрах удою молока на добу за попередній аналогічний прогнозуванню період;

– середня тривалість румінації свійської тварини в годинах на добу;

– наявність вакцини проти сказу;

– наявність вакцини проти сибірської виразки;

– наявність вакцини проти бруцельозу;

– наявність вакцини проти ящура;

– наявність вакцини проти герпесу;

– наявність вакцини проти вірусної діареї;

– наявність вакцини проти маститу;

– наявність вакцини проти інфекційного ринотрахеїту;

– середня кількість в літрах удою молока на добу за наступний період.

Під час дослідження було використано метод опорних векторів [24]-[25] та дерева рішень [5]-[7] у складі запропонованого в роботі методу. Виконано прогнозування обсягу удоїв молока худоби.

Оцінювання результатів відбувалось на основі таких показників:

– середня абсолютна похибка (САП);

– середньоквадратична похибка (СКП);

– кореневе середньоквадратичне відхилення (КСКВ).

40 % вибірки даних було віднесено до тестової вибірки, на якій було отримано результати експериментів (табл. 5.1). 60 % було віднесено до навчальної вибірки.

Таблиця 5.1 – Результати прогнозування удою худоби

Метод / модель	САП	КСКВ	СКП
Метод опорних векторів	0,0019092259	0,0021246919	0,0000045143
Дерево рішень (запропонований метод)	0,0003046144	0,0003837075	0,0000001472

Запропонований метод на основі дерев рішень призвів до зменшення значення САП порівняно з методом опорних векторів у 6,27 разів, до зменшення значення КСКВ у 5,54 разів, до зменшення значення СКП у 30,67 разів.

5.4 Висновки за розділом 5

Сукупність сценаріїв роботи користувача з програмою відслідковування стану худоби продемонстрована у даному розділі з визначенням логіки доступу до таких функцій. Проведено експериментальне дослідження прогнозування обсягу удоїв молока худоби. Запропонований метод дозволив зменшити рівень похибки САП, КСКВ, СКП порівняно з альтернативою у вигляді методу опорних векторів.

ВИСНОВКИ

У роботі повністю досягнуто мету, що передбачала збільшення точності прогнозування стану худоби шляхом розробки програмних засобів відслідковування стану худоби. Збільшення точності прогнозування стану худоби забезпечено створенням і навчанням моделі прогнозування на основі запропонованого методу, програмною реалізацією такої моделі та її застосуванням у відповідних сценаріях для роботи з користувачем на основі зібраних даних про стан худоби.

Сукупність засобів для розробки програми відслідковування стану худоби за результатами виконання третього розділу включає мову Python, фреймворк Django для забезпечення підтримки безпосередньо створення вебзастосунку, системи MySQL для організації бази даних. Моделювання сценаріїв забезпечено для користувача програми відслідковування стану худоби зі створенням діаграми прецедентів. Створено відповідну структуру бази даних, побудовано її схему.

Створена програма відслідковування стану худоби необхідна користувачам для керування даними худоби за тим господарством, яке вони представляють, при цьому контролюючи цілу сукупність станів, дій, які пов’язані зі щоденною роботою з худобою. До цього входять загальні дані, дані вакцинацій, хвороб, фактичних і прогнозованих зборів молока, переходів на пасовиська, вимірювання стану свійської тварини, народження, продажу або придбання тощо.

Наукова новизна роботи полягає в методі прогнозування стану худоби, що характеризується використанням дерев рішень, визначенням обчислення вхідних ознак, що дозволяє визначати обсяг удоїв худоби на наступний період.

Проведено експериментальне дослідження прогнозування обсягу удоїв молока худоби. Запропонований метод дозволив зменшити рівень похибки САП, КСКВ, СКП прогнозування стану худоби у вигляді обсягу удоїв молока порівняно з альтернативою у вигляді методу опорних векторів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. FarmKeep. Modern comprehensive farm management app [Electronic resource]. – Access mode : https://www.farmkeep.com/.

2. FarmKeep [Electronic resource] : Farm & Homestead App – Apps on Google Play. – Access mode : https://play.google.com/store/apps/details?id=com.farmkeep.farmkeep&pcampaignid=web_share.

3. Manage Livestock Farms Effortlessly with Cattlytics Software [Electronic resource]. – Access mode : https://www.cattlytics.com/.

4. Cattle Lifecycle Tracking and Record Keeping Management [Electronic resource]. – Access mode : https://www.cattlytics.com/features/cattle-lifecycle-records-management/.

5. What Is a Decision Tree? Master's in Data Science [Electronic resource]. – Access mode : https://www.mastersindatascience.org/learning/machine-learning-algorithms/decision-tree/.

6. Smith, C. Decision Trees and Random Forests [Text] : A Visual Introduction For Beginners / C. Smith. – Vancouver : Blue Windmill Media, 2017. – 168 p.

7. Sheppard, C. Tree-based Machine Learning Algorithms [Text] : Decision Trees, Random Forests, and Boosting / C. Sheppard. – CreateSpace Independent Publishing Platform, 2017. – 110 p.

8. What is Python? – Python Language Explained – AWS [Electronic resource]. – Access mode : https://aws.amazon.com/what-is/python/.

9. Python Applications and Advantages [Electronic resource]. – Access mode : https://www.geeksforgeeks.org/python/python-language-advantages-applications/.

10. Bader, D. Python Tricks [Text] : A Buffet of Awesome Python Features / D. Bader. – Vancouver : Dan Bader (dbader.org), 2017. – 326 p.

11. Ramalho, L. Fluent Python [Text] : Clear, Concise, and Effective Programming / L. Ramalho. – Sebastopol, CA : O'Reilly Media, 2022. – 1012 p.

12. Беррі, П. Head First Python. Легкий для сприйняття довідник [Текст] / П. Беррі. – К. : Фабула, 2021. – 624 с.

13. McGavren, J. Head First Ruby. A Brain Friendly Guide [Text] / J. McGavren. – Boston : O'Reilly Media, 2015. – 569 p.

14. Ruby, S. Agile Web Development with Rails 8 [Text] / S. Ruby, D. Thomas. – Boston : Pragmatic Bookshelf, 2025. – 475 p.

15. Elder, J. Learn Ruby On Rails For Web Development [Text] : Learn Rails The Fast And Easy Way! / J. Elder. – Chicago : Codemy.com, 2015. – 214 p.

16. Hartl, M. Ruby on Rails Tutorial [Text] : Learn Web Development with Rails (Addison-Wesley Professional Ruby Series) / M. Hartl. – Boston : Addison-Wesley Professional, 2023. – 896 p.

17. What is Django? – Django Software Explained – AWS [Electronic resource]. – Access mode : https://aws.amazon.com/what-is/django/.

18. Django Tutorial [Electronic resource] : Python Web Development | DataCamp. – Access mode : https://www.datacamp.com/tutorial/web-development-django.

19. Mele, A. Django 5 By Example [Text] : Build powerful and reliable Python web applications from scratch / A. Mele. – Birmingham : Packt Publishing, 2024. – 820 p.

20. Free Template 602 Graph Page [Electronic resource]. – Access mode : https://templatemo.com/tm-602-graph-page.

21. Hackeling, G. Mastering Machine Learning with scikit-learn – Second Edition: Apply effective learning algorithms to real-world problems using scikit-learn [Text] / G. Hackeling. – Birmingham : Packt Publishing, 2017. – 254 p.

22. Geron, A. Hands-On Machine Learning with Scikit-Learn and PyTorch [Text] : Concepts, Tools, and Techniques to Build Intelligent Systems / A. Geron. – Santa Rosa : O'Reilly Media, 2025. – 875 p.

23. Cattle Health and Feeding Data [Electronic resource]. – Access mode : https://www.kaggle.com/datasets/shahhet2812/cattle-health-and-feeding-data.

24. Support Vector Machine (SVM) Algorithm [Electronic resource]. –

Access mode : https://www.mygreatlearning.com/blog/introduction-to-support-vector-machine/.

25. Vijayan, V. Unveiling the Power of Support Vector Machines in Machine Learning [Electronic resource] / V. Vijayan. – Access mode : https://dev.to/vjnvisakh/unveiling-the-power-of-support-vector-machines-in-machine-learning-2p32.

ДОДАТОК А

Технічне завдання

Вступ

Робота з худобою господарства є доволі кропіткою, а у випадку наявності великої кількості худоби контроль за станом такої худоби стає проблематичним без використання програмних засобів. Програмні засоби дозволяють розуміти поточний стан худоби, виходячи з історії спостережень за нею. З іншого боку ці ж дані можуть бути не тільки історією стосовно худоби, але і тими даними, на основі яких можна побудувати прогноз. Тобто розуміючи загальний стан худоби, робити прогноз щодо того, яку кількість молока вдасться зібрати в такої тварини. Це дозволяє планувати діяльність господарства, розуміючи, скільки продукції буде отримано. У випадку проблем зі здоров'ям тварини необхідно мати можливість планувати подальшу роботу, орієнтуючись на близькі до подальших фактичних результатів. Тому інтеграція інструментів прогнозування зокрема удоїв молока худоби дозволить мати комплексний програмний засіб для відслідковування стану худоби господарства, стаючи основою для подальшого планування роботи.

А.1 Підстави для розробки

Наказ № 447 від 30.09.2025 року за Національним університетом «Запорізька політехніка» використовується для даної дипломної кваліфікаційної роботи як підстава для розробки за темою «Дослідження та програмна реалізація методів відслідковування стану худоби».

А.2 Призначення розробки

Програма відслідковування стану худоби необхідна користувачам для керування даними худоби за тим господарством, яке вони представляють, при цьому контролюючи цілу сукупність станів, дій, які пов'язані зі щоденною

роботою з худобою. До цього входять загальні дані, дані вакцинацій, хвороб, фактичних і прогнозованих зборів молока, переходів на пасовиська, вимірювання стану свійської тварини, народження, продажу або придбання тощо.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Функціональні характеристики програмного забезпечення відслідковування стану худоби мають включати:

– перегляд даних власного господарства;

– авторизація і реєстрація;

– перегляд користувачів, які представляють господарство;

– надання права працювати від імені господарства користувачу та скасування права;

– редагування даних господарства;

– додавання приміщення до господарства;

– перегляд всіх приміщень господарства;

– редагування даних приміщення господарства;

– вилучення приміщення з господарства;

– перегляд всієї худоби, зареєстрованої за господарством;

– реєстрація нової худоби господарством;

– редагування даних зареєстрованої худоби;

– зняття худоби з обліку;

– реєстрація продажу худоби;

– перегляд даних приміщення господарства;

– реєстрація придбання існуючої в програмі худоби;

– перегляд всієї придбаної господарством худоби;

– перегляд всієї проданої господарством худоби;

– перегляд поголів’я народженої худоби в господарстві;

– перегляд поголів’я народженої твариною худоби;

– перегляд динаміки народження худоби за господарством;

– реєстрація народження худоби;

– редагування народження худоби;

– вилучення даних народження худоби;

– внесення удою худоби за твариною;

– внесення удою худоби за породою тварин;

– внесення удою худоби за приміщенням господарства;

– перегляд динаміки удоїв худоби загалом за господарством;

– перегляд динаміки удоїв худоби заданої породи за господарством;

– перегляд динаміки удоїв конкретної тварини;

– перегляд історії удоїв худоби за приміщенням господарства;

– редагування внесеного удою худоби;

– вилучення внесеного удою худоби;

– прогнозування удою худоби;

– встановлення базового варіанта годування худоби;

– встановлення параметрів годування худоби;

– внесення годування худоби за твариною;

– внесення годування худоби за породою тварин;

– внесення годування худоби за приміщенням господарства;

– перегляд історії годування худоби за приміщенням господарства;

– перегляд історії годування худоби за твариною;

– редагування внесеного годування худоби;

– вилучення внесеного годування худоби;

– вилучення невикористаного базового варіанта годування худоби;

– редагування невикористаного базового варіанта годування худоби;

– редагування невикористаних параметрів годування худоби;

– вилучення невикористаних параметрів годування худоби;

– перегляд історії зміни варіантів годування худоби за твариною;

– перегляд історії зміни параметрів годування худоби за твариною;

– перегляд історії зміни варіантів годування худоби в господарстві за породою;

– перегляд історії зміни параметрів годування худоби в господарстві за породою;

– перегляд історії зміни обсягів годування худоби в господарстві за варіантом годування;

– внесення дій над худобою за твариною;

– внесення дій над худобою за породою тварин;

– внесення дій над худобою за приміщенням господарства;

– редагування внесеної дії за худобою;

– визначення завершення дії за худобою;

– вилучення внесеної дії за худобою;

– перегляд дій над худобою за твариною;

– перегляд дій над худобою за приміщенням господарства;

– перегляд активності дій над худобою за господарством;

– внесення вимірювань стану худоби за твариною;

– внесення вимірювань стану худоби за приміщенням господарства;

– внесення вимірювань стану худоби за породою;

– редагування внесеного вимірювання стану худоби;

– вилучення внесеного вимірювання стану худоби;

– перегляд історії зміни стану худоби за твариною;

– перегляд динаміки зміни стану худоби за приміщенням господарства;

– перегляд динаміки зміни стану худоби за породою;

– внесення вакцинації худоби за твариною;

– внесення вакцинації худоби за приміщенням господарства;

– внесення вакцинації худоби за породою;

– перегляд історії вакцинації худоби за твариною;

– перегляд стану вакцинації худоби за приміщенням господарства;

– перегляд стану вакцинації худоби за породою;

– вилучення внесеної вакцинації худоби;

– внесення хвороби худоби;

– визначення завершення хвороби худоби;

– вилучення внесеної хвороби тварини;

– перегляд історії захворювання худоби за твариною;

– перегляд історії захворювання худоби за приміщенням господарства;

– перегляд історії захворювання худоби за породою;

– перегляд динаміки захворювання худоби за господарством;

– перегляд динаміки захворювання худоби за приміщенням господарства;

– перегляд динаміки захворювання худоби за породою;

– пошук худоби за власним господарством;

– пошук худоби за іншим господарством;

– підтвердження продажу худоби;

– скасування даних продажу худоби;

– перегляд поданих заявок придбань худоби господарства;

– перегляд власних поданих заявок придбань худоби;

– вилучення даних продажу худоби;

– внесення погодних умов за господарством;

– редагування внесених погодних умов за господарством;

– перегляд внесених погодних умов за господарством.

А.3.2 Умови експлуатації

Для запуску програмного забезпечення відслідковування стану худоби на сервері (на відміну від клієнта, де наявний сучасний браузер та підключення до інтернет є достатніми) потрібно мати:

– сучасний вебсервер (наприклад, Apache);

– засоби підтримки розробки у відповідності з вибором, який має бути зроблений в роботі.

А.3.3 Вимоги до складу та параметрів технічних засобів

Сервер підтримки програмного забезпечення відслідковування стану худоби повинен мати такі технічні засоби:

– багатоядерний процесор з частотою щонайменше 3 ГГц;

– 8 Гб оперативної пам’яті мінімально для підтримки повноцінної роботи зі створенням моделі прогнозування;

– 1 Гб вільного для використання програми відслідковування стану худоби обсягу жорсткого диску.

А.4 Порядок контролю та приймання

Календарний план на дипломну кваліфікаційну роботу, що є частиною завдання, містить вимоги до контролю роботи з поетапним виконанням. Приймання роботи відбувається керівником, нормоконтролерами, завідувачем кафедри та в підсумку екзаменаційною комісією під час захисту.

ДОДАТОК Б

Текст програми

```
      from django.shortcuts import render
      from cattlemoni.models import Cattleregion, Cattlecountry,
Cattlebreed, Locationplace, Household, Barn, Cattle, Cattlesale,
Cattlebirth, Cattlemilking, Cattleprediction,
Cattlefeedbasetype, Cattlefeedtype, Cattlefeedbase, Cattlefeed,
Cattleactivity, Cattlewalking, Cattlemeasurement,
Ambientweather, Cattlevaccine, Diseasetype, Cattledisease,
VACCINE_TYPES
      from django.core.paginator import Paginator
      import datetime

      def household_view(request):
          household_results = {}
          if request.user.is_authenticated:
              user_household =
Household.objects.select_related("locationplace",

"locationplace__cattlecoutry",

"locationplace__cattlecoutry__cregion",

"users").\
                  filter(users__in = [request.user])
              if user_household:
                  user_household = user_household[0]
                  household_results["household"] = user_household
                  household_cattles =
Cattle.objects.filter(barnlocation__household = user_household,

state__in = ["A", "D"]).\
                                          order_by("-
fullbirthdate")[:5]
                  household_results["cattles"] =
household_cattles
                  household_cattles_count =
household_cattles.count()
                  household_results["cattles_count"] =
household_cattles_count
                  household_barns = Barn.objects.filter(household
= user_household).\
                      order_by("-barnscore")
                  household_barns_count = household_barns.count()
                  household_results["barns"] =
household_barns[:4]
                  household_results["barns_count"] =
household_barns_count
                  household_cattle_breeds =
Cattle.objects.filter(barnlocation__household = user_household,

state__in = ["A", "D"]).\
                      values('cattlebreed__id').distinct()
                  household_cattle_breeds =
Cattlebreed.objects.filter(id__in = household_cattle_breeds)
```

```
                    for household_cattle_breed in
household_cattle_breeds:
                        household_cattle_breed.count =
Cattle.objects.filter(barnlocation__household = user_household,

state__in = ["A", "D"],

cattlebreed = household_cattle_breed).count()
                    household_results["breeds_count"] =
household_cattle_breeds.count()
                    household_results["breeds"] =
household_cattle_breeds[:5]
                    cnow = datetime.datetime.now()
                    cmonth = cnow.month
                    cyear = cnow.year
                    firstdate = datetime.date(day = 1, month =
cmonth, year = cyear)
                    if cmonth < 12:
                        lastdate = datetime.date(day=1,
month=cmonth + 1, year=cyear) - datetime.timedelta(days=1)
                    else:
                        lastdate = datetime.date(day=1, month=1,
year=cyear+1) - datetime.timedelta(days=1)
                    household_measurements =
Cattlemeasurement.objects.select_related("cattle").filter(measur
etime__gte = firstdate,

measuretime__lte = lastdate,

cattle__barnlocation__household = user_household).\
                        order_by("-measuretime")
                    household_measurements_count =
household_measurements.count()
                    household_measurements =
household_measurements[:5]
                    household_results["measurements_count"] =
household_measurements_count
                    household_results["measurements"] =
household_measurements
                    if household_measurements:
                        household_results["measurements_lastdate"]
= household_measurements[0]
                        household_results["measurements_lastdate"]
= household_results["measurements_lastdate"].measuretime.date()
                    yearlongdate = cnow -
datetime.timedelta(days=365)
                    household_results["disease_active_count"] =
Cattledisease.objects.filter(diseaseend__isnull = True,

diseasestart__gte = yearlongdate,

cattle__barnlocation__household = user_household).\
                        count()
```

```
                household_results["diseases"] =
Cattledisease.objects.all().order_by("-diseasestart")[:5]
                vaccines = []
                for vaccine_type in VACCINE_TYPES.keys():
                    vaccine = (VACCINE_TYPES[vaccine_type],

Cattlevaccine.objects.select_related("cattle").filter(activetill
__gte = cnow,

cattle__barnlocation__household = user_household,

vaccinetype = vaccine_type).\
                                count())
                    vaccines.append(vaccine)
                household_results["vaccines"] = vaccines
                household_results["vaccines_lastdate"] =
Cattlevaccine.objects.filter(cattle__barnlocation__household =
user_household).\
                    order_by("-vaccinetime")
                if household_results["vaccines_lastdate"]:
                    household_results["vaccines_lastdate"] =
household_results["vaccines_lastdate"][0]
                    household_results["vaccines_lastdate"] =
household_results["vaccines_lastdate"].vaccinetime
                ambientweather_set =
Ambientweather.objects.filter(household=user_household).\
                                            order_by("-
ambienttime")[:5]
                if ambientweather_set:
                    household_results["weather_lastdate"] =
ambientweather_set[0].date()
                household_results["weathers"] =
ambientweather_set
                thirty_days_ago = cnow -
datetime.timedelta(days=30)
                month_milkings =
Cattlemilking.objects.select_related("cattle").filter(cattle__ba
rnlocation__household = user_household,

milkingtime__gte = thirty_days_ago).\
                    order_by("-milkingtime")
                household_results["milkings"] =
month_milkings[:5]
                if household_results["milkings"]:
                    household_results["milking_lastdate"] =
month_milkings[0].milkingtime
                household_results["milkings_30days_literes"] =
sum(month_milkings.values_list("milkvolume"))
                month_milkings =
Cattlemilking.objects.select_related("cattle").filter(cattle__ba
rnlocation__household=user_household,

milkingtime__gte=firstdate)
```

```
                household_results["milkings_month_literes"] =
sum(month_milkings.values_list("milkvolume"))
                household_results["activities"] =
Cattleactivity.objects.select_related("cattle").filter(cattle__b
arnlocation__household=user_household).\
                    order_by("-activitystarttime")[:5]
                if household_results["activities"]:
                    household_results["activity_lastdate"] =
household_results["activities"][0].date()
                household_results["activity_todo_number"] =
Cattleactivity.objects.select_related("cattle").filter(cattle__b
arnlocation__household=user_household,

activityendtime__isnull = True,

activitystarttime = cnow.date()).\
                    count()
                household_results["feeds"] =
Cattlefeed.objects.select_related("cattlefeedbase",

"cattlefeedbase__cattlefeedtype",

"cattlefeedbase__cattlefeedtype__cattle").filter(

cattlefeedbase__cattlefeedtype__cattle__barnlocation__household=
user_household). \

order_by("-feedtime")[:5]
                if household_results["feeds"]:
                    household_results["feed_lastdate"] =
household_results["feeds"][0].date()
        return render(request, "household.html",
household_results)

    def household_barns_view(request, page = 1):
        household_barns_results = {}
        if request.user.is_authenticated:
            user_household =
Household.objects.select_related("locationplace",

"locationplace__cattlecoutry",

"locationplace__cattlecoutry__cregion",

"users").\
                    filter(users__in = [request.user])
            user_household = user_household[0]
            household_barns_results["household"] =
user_household
            household_barns = Barn.objects.filter(household =
user_household).\
                    order_by("-barnscore")
            household_barns_count = household_barns.count()
```

```
            household_barns_paginator =
Paginator(household_barns, 10)
            household_barns =
household_barns_paginator.get_page(page)
            household_barns_results["barns"] = household_barns
            household_barns_results["barns_count"] =
household_barns_count
        return render(request, "householdbarns.html",
household_barns_results)

    def household_cattles_view(request, page = 1, barn_id = -1,
cattle_breed_id = -1):
        household_cattles_results = {}
        if request.user.is_authenticated:
            user_household =
Household.objects.select_related("locationplace",

"locationplace__cattlecoutry",

"locationplace__cattlecoutry__cregion",

"users").\
                filter(users__in = [request.user])
            user_household = user_household[0]
            household_cattles_results["household"] =
user_household
            if barn_id == -1 and cattle_breed_id == -1:
                household_cattles =
Cattle.objects.filter(barnlocation__household = user_household,
                                                    state__in
= ["A", "D"]).\
                                        order_by("-fullbirthdate")
            elif barn_id != -1 and cattle_breed_id == -1:
                household_cattles =
Cattle.objects.filter(barnlocation__household=user_household,

barnlocation__id = barn_id,

state__in=["A", "D"]). \
                    order_by("-fullbirthdate")
                household_cattles_results["barn"] =
Barn.objects.get(id = barn_id)
            elif barn_id == -1 and cattle_breed_id != -1:
                household_cattles =
Cattle.objects.filter(barnlocation__household=user_household,

cattlebreed__id=cattle_breed_id,

state__in=["A", "D"]). \
                    order_by("-fullbirthdate")
                household_cattles_results["breed"] =
Cattlebreed.objects.get(id=cattle_breed_id)
            else:
```

```
                household_cattles =
Cattle.objects.filter(barnlocation__household=user_household,

cattlebreed__id=cattle_breed_id,

barnlocation__id=barn_id,

state__in=["A", "D"]). \
                    order_by("-fullbirthdate")
                household_cattles_results["barn"] =
Barn.objects.get(id=barn_id)
                household_cattles_results["breed"] =
Cattlebreed.objects.get(id=cattle_breed_id)
            household_cattles_count = household_cattles.count()
            household_cattles_results["cattles_count"] =
household_cattles_count
            household_cattles_paginator =
Paginator(household_cattles, 10)
            household_cattles =
household_cattles_paginator.get_page(page)
            household_cattles_results["cattles"] =
household_cattles
            household_barns = Barn.objects.filter(household =
user_household).\
                order_by("barnname")
            household_cattles_results["barns"] =
household_barns
            household_cattle_breeds =
Cattle.objects.filter(barnlocation__household = user_household,

state__in = ["A", "D"]).\
                values('cattlebreed__id').distinct()
            household_cattle_breeds =
Cattlebreed.objects.filter(id__in = household_cattle_breeds)
            household_cattles_results["breeds"] =
household_cattle_breeds
        return render(request, "householdcattles.html",
household_cattles_results)

    def household_barn_vaccination_view(request, barn_id = -1):
        household_barn_vaccination_results = {}
        household_barn_vaccination_results["barns"] =
Barn.objects.all().order_by("barnname")
        if request.user.is_authenticated:
            user_household =
Household.objects.select_related("locationplace",

"locationplace__cattlecoutry",

"locationplace__cattlecoutry__cregion",

"users").\
                filter(users__in = [request.user])
```

```
            user_household = user_household[0]
            household_barn_vaccination_results["household"] =
user_household
            if barn_id == -1:
                household_cattles =
Cattle.objects.filter(barnlocation__household=user_household,

state__in=["A", "D"]). \
                      order_by("-fullbirthdate")
            else:
                household_cattles =
Cattle.objects.filter(barnlocation__household=user_household,

barnlocation__id=barn_id,

state__in=["A", "D"]). \
                      order_by("-fullbirthdate")
                household_barn_vaccination_results["barn"] =
Barn.objects.get(id=barn_id)
            household_barn_vaccination_results["cattles"] =
household_cattles
            household_cattles_count = household_cattles.count()
            household_barn_vaccination_results["cattles_count"]
= household_cattles_count
            household_barns = Barn.objects.filter(household =
user_household).\
                order_by("-barnscore")
            household_barns_count = household_barns.count()
            household_barn_vaccination_results["barns"] =
household_barns
            household_barn_vaccination_results["barns_count"] =
household_barns_count
            cnow = datetime.datetime.now()
            vaccines = []
            for vaccine_type in VACCINE_TYPES.keys():
                for cattle in
household_barn_vaccination_results["cattles"]:

vaccines.append((VACCINE_TYPES[vaccine_type],
                         cattle,

Cattlevaccine.objects.select_related("cattle").filter(activetill
__lte = cnow,

cattle = cattle,

vaccinetype = vaccine_type)))
            household_barn_vaccination_results["vaccines"] =
vaccines

household_barn_vaccination_results["vaccines_lastdate"] =
Cattlevaccine.objects.filter(cattle__barnlocation__household =
user_household).\
```

```
                order_by("-vaccinetime")[0]

household_barn_vaccination_results["vaccines_lastdate"] =
household_barn_vaccination_results["vaccines_lastdate"].vaccinet
ime
            vaccines = []
            for vaccine_type in VACCINE_TYPES.keys():
                vaccines.append((VACCINE_TYPES[vaccine_type],

Cattlevaccine.objects.select_related("cattle").filter(activetill
__lte=cnow,

cattle__barnlocation__household=user_household,

vaccinetype=vaccine_type). \
                        count()))

household_barn_vaccination_results["generalvaccines"] = vaccines
        return render(request, "barnvaccinestate.html",
household_barn_vaccination_results)

    def household_breed_vaccination_view(request,
cattle_breed_id = -1):
        household_breed_vaccination_results = {}
        household_breed_vaccination_results["breeds"] =
Cattlebreed.objects.all().order_by("cbreedname")
        if request.user.is_authenticated:
            user_household =
Household.objects.select_related("locationplace",

"locationplace__cattlecoutry",

"locationplace__cattlecoutry__cregion",

"users").\
                filter(users__in = [request.user])
            user_household = user_household[0]
            household_breed_vaccination_results["household"] =
user_household
            if cattle_breed_id == -1:
                household_cattles =
Cattle.objects.filter(barnlocation__household=user_household,

state__in=["A", "D"]). \
                    order_by("-fullbirthdate")
            else:
                household_cattles =
Cattle.objects.filter(barnlocation__household=user_household,

cattlebreed__id=cattle_breed_id,

state__in=["A", "D"]). \
                    order_by("-fullbirthdate")
```

```
                household_breed_vaccination_results["breed"] =
Cattlebreed.objects.get(id=cattle_breed_id)
            household_breed_vaccination_results["cattles"] =
household_cattles
            household_cattles_count = household_cattles.count()

household_breed_vaccination_results["cattles_count"] =
household_cattles_count
            household_barns = Barn.objects.filter(household =
user_household).\
                order_by("-barnscore")
            household_barns_count = household_barns.count()
            household_breed_vaccination_results["barns"] =
household_barns
            household_breed_vaccination_results["barns_count"]
= household_barns_count
            cnow = datetime.datetime.now()
            vaccines = []
            for cattle in
household_breed_vaccination_results["cattles"]:
                vac = []
                vac.append(cattle)
                for vaccine_type in VACCINE_TYPES.keys():
                    res =
Cattlevaccine.objects.select_related("cattle").filter(activetill
__gte=cnow,

cattle=cattle,

vaccinetype=vaccine_type)
                    if res:
                        vac.append(res[0])
                    else:
                        vac.append(False)
                vaccines.append(tuple(vac))
            household_breed_vaccination_results["vaccines"] =
vaccines

household_breed_vaccination_results["vaccines_lastdate"] =
Cattlevaccine.objects.filter(cattle__barnlocation__household =
user_household).\
                order_by("-vaccinetime")[0]

household_breed_vaccination_results["vaccines_lastdate"] =
household_breed_vaccination_results["vaccines_lastdate"].vaccine
time
            vaccines = []
            for vaccine_type in VACCINE_TYPES.keys():
                vaccines.append((VACCINE_TYPES[vaccine_type],

Cattlevaccine.objects.select_related("cattle").filter(activetill
__gte=cnow,
```

```
                                    cattle__barnlocation__household
=user_household,

vaccinetype=vaccine_type). \
                           count()))

household_breed_vaccination_results["generalvaccines"] =
vaccines
          return render(request, "breedvaccinestate.html",
household_breed_vaccination_results)

      def cattle_vaccination_view(request, cattle_id, page = 1):
          cattle_vaccination_results = {}
          if request.user.is_authenticated:
              user_household =
Household.objects.select_related("locationplace",

"locationplace__cattlecoutry",

"locationplace__cattlecoutry__cregion",

"users").\
                  filter(users__in = [request.user])
              user_household = user_household[0]
              cattle_vaccination_results["household"] =
user_household
              household_cattle =
Cattle.objects.select_related("cattlebreed", "barnlocation").\
                  get(barnlocation__household=user_household,
                                                      id =
cattle_id)
              cattle_vaccination_results["cattle"] =
household_cattle
              cnow = datetime.datetime.now()
              vaccines = []
              for vaccine_type in VACCINE_TYPES.keys():
                  vaccines.append((VACCINE_TYPES[vaccine_type],

Cattlevaccine.objects.select_related("cattle").filter(activetill
__lte = cnow,

cattle = household_cattle,

vaccinetype = vaccine_type)))
              cattle_vaccination_results["vaccinestates"] =
vaccines
              vaccines = []
              for vaccine_type in VACCINE_TYPES.keys():
                  res =
Cattlevaccine.objects.select_related("cattle").filter(activetill
__lte=cnow,

cattle__barnlocation__household=user_household,
```

```
                                                        vaccinetype
=vaccine_type). \
                        count()
                    vaccines.append(res)
                cattle_vaccination_results["generalvaccines"] =
vaccines
                vaccines =
Cattlevaccine.objects.select_related("cattle").filter(cattle=hou
sehold_cattle).\
                    order_by("-vaccinetime")
                cattle_vaccine_paginator = Paginator(vaccines, 10)
                vaccines = cattle_vaccine_paginator.get_page(page)
                cattle_vaccination_results["vaccinehistory"] =
vaccines
            return render(request, "cattlevaccinestate.html",
cattle_vaccination_results)

        def cattle_diseases_view(request, cattle_id, page = 1):
            cattle_disease_results = {}
            if request.user.is_authenticated:
                user_household =
Household.objects.select_related("locationplace",

"locationplace__cattlecoutry",

"locationplace__cattlecoutry__cregion",

"users").\
                    filter(users__in = [request.user])
                user_household = user_household[0]
                cattle_disease_results["household"] =
user_household
                household_cattle =
Cattle.objects.select_related("cattlebreed", "barnlocation").\
                    get(barnlocation__household=user_household,
                                                        id =
cattle_id)
                cattle_disease_results["cattle"] = household_cattle
                cnow = datetime.datetime.now()
                yearlongdate = cnow - datetime.timedelta(days=365)
                cattle_disease_results["diseasestate"] =
Cattledisease.objects.filter(diseaseend__isnull=True,

diseasestart__gte=yearlongdate,

cattle__barnlocation__household=user_household).\
                    count()
                diseases =
Diseasetype.objects.all().order_by("diseasename")
                cattle_disease_results["diseases"] = diseases
                cattle_diseases = []
                histories = []
                for disease in diseases:
```

```
                disease_general_state =
Cattledisease.objects.filter(diseaseend__isnull=True,

cattle__barnlocation__household=user_household,

disease = disease).\
                    count()
                cattle_diseases_history =
Cattledisease.objects.filter(cattle=household_cattle,

disease=disease). \
                    order_by("-diseasestart")
                cattle_history_paginator =
Paginator(cattle_diseases_history, 10)
                cattle_diseases_history =
cattle_history_paginator.get_page(page)
                cattle_diseases.append(disease_general_state)
                histories.append(cattle_diseases_history)
            cattle_disease_results["diseasehistory"] =
cattle_diseases
            cattle_disease_results["histories"] = histories
        return render(request, "cattlediseasehistory.html",
cattle_disease_results)

    def barn_diseases_view(request, barn_id = -1, page = 1):
        barn_disease_results = {}
        barn_disease_results["barns"] =
Barn.objects.all().order_by("barnname")
        if request.user.is_authenticated:
            user_household =
Household.objects.select_related("locationplace",

"locationplace__cattlecoutry",

"locationplace__cattlecoutry__cregion",

"users").\
                    filter(users__in = [request.user])
            user_household = user_household[0]
            barn_disease_results["household"] = user_household
            if barn_id == -1:
                household_cattles =
Cattle.objects.filter(barnlocation__household=user_household,

state__in=["A", "D"]). \
                    order_by("-fullbirthdate")
            else:
                household_cattles =
Cattle.objects.filter(barnlocation__household=user_household,

barnlocation__id=barn_id,

state__in=["A", "D"]). \
```

```
                    order_by("-fullbirthdate")
                barn_disease_results["barn"] =
Barn.objects.get(id=barn_id)
            barn_disease_results["cattles"] = household_cattles
            cnow = datetime.datetime.now()
            yearlongdate = cnow - datetime.timedelta(days=365)
            barn_disease_results["diseasestate"] =
Cattledisease.objects.filter(diseaseend__isnull=True,

diseasestart__gte=yearlongdate,

cattle__barnlocation__household=user_household).\
                count()
            diseases =
Diseasetype.objects.all().order_by("diseasename")
            barn_disease_results["diseases"] = diseases
            cattle_diseases = []
            general_cattle_diseases = []
            cattle_diseases_histories = []
            for disease in diseases:
                disease_general_state =
Cattledisease.objects.filter(diseaseend__isnull=True,

cattle__barnlocation__household=user_household,

disease = disease).\
                    count()
                disease_barn_state =
Cattledisease.objects.filter(diseaseend__isnull=True,

cattle__in=household_cattles,

disease=disease). \
                    count()
                cattle_diseases_history =
Cattledisease.objects.filter(cattle__in = household_cattles,

disease = disease).\
                    order_by("-diseasestart")
                cattle_history_paginator =
Paginator(cattle_diseases_history, 10)
                cattle_diseases_history =
cattle_history_paginator.get_page(page)
                cattle_diseases.append(disease_general_state)

general_cattle_diseases.append(disease_barn_state)

cattle_diseases_histories.append(cattle_diseases_history)
            barn_disease_results["diseasehistory"] =
cattle_diseases
            barn_disease_results["barndiseasehistory"] =
general_cattle_diseases
```

```
            barn_disease_results["histories"] =
cattle_diseases_histories
        return render(request, "barndiseasehistory.html",
barn_disease_results)

    def breed_diseases_view(request, cattle_breed_id = -1, page
= 1):
        breed_disease_results = {}
        breed_disease_results["breeds"] =
Cattlebreed.objects.all().order_by("cbreedname")
        if request.user.is_authenticated:
            user_household =
Household.objects.select_related("locationplace",

"locationplace__cattlecoutry",

"locationplace__cattlecoutry__cregion",

"users").\
                filter(users__in = [request.user])
            user_household = user_household[0]
            breed_disease_results["household"] = user_household
            if cattle_breed_id == -1:
                household_cattles =
Cattle.objects.filter(barnlocation__household=user_household,

state__in=["A", "D"]). \
                    order_by("-fullbirthdate")
            else:
                household_cattles =
Cattle.objects.filter(barnlocation__household=user_household,

cattlebreed__id=cattle_breed_id,

state__in=["A", "D"]). \
                    order_by("-fullbirthdate")
                breed_disease_results["breed"] =
Cattlebreed.objects.get(id=cattle_breed_id)
            breed_disease_results["cattles"] =
household_cattles
            cnow = datetime.datetime.now()
            yearlongdate = cnow - datetime.timedelta(days=365)
            breed_disease_results["diseasestate"] =
Cattledisease.objects.filter(diseaseend__isnull=True,

diseasestart__gte=yearlongdate,

cattle__barnlocation__household=user_household).\
                count()
            diseases =
Diseasetype.objects.all().order_by("diseasename")
            breed_disease_results["diseases"] = diseases
            cattle_diseases = []
```

```
            general_cattle_diseases = []
            cattle_diseases_histories = []
            for disease in diseases:
                disease_general_state =
Cattledisease.objects.filter(diseaseend__isnull=True,

cattle__barnlocation__household=user_household,

disease = disease).\
                    count()
                disease_breed_state =
Cattledisease.objects.filter(diseaseend__isnull=True,

cattle__in=household_cattles,

disease=disease). \
                    count()
                cattle_diseases_history =
Cattledisease.objects.filter(cattle__in = household_cattles,

disease = disease).\
                    order_by("-diseasestart")
                cattle_history_paginator =
Paginator(cattle_diseases_history, 10)
                cattle_diseases_history =
cattle_history_paginator.get_page(page)
                cattle_diseases.append(disease_general_state)

general_cattle_diseases.append(disease_breed_state)

cattle_diseases_histories.append(cattle_diseases_history)
            breed_disease_results["diseasehistory"] =
cattle_diseases
            breed_disease_results["breeddiseasehistory"] =
general_cattle_diseases
            breed_disease_results["histories"] =
cattle_diseases_histories
        return render(request, "breeddiseasehistory.html",
breed_disease_results)

    from django.db.models import Q
    from dateutil.relativedelta import relativedelta

    def barn_disease_dynamics_view(request, barn_id = -1,
startdate = None, enddate = None):
        if startdate:
            startdates = startdate.split('-')
            startdate = datetime.datetime.date(year =
int(startdates[0]), month = int(startdates[1]), day =
int(startdates[2]))
        if enddate:
            enddates = enddate.split('-')
```

```
            enddate = datetime.datetime.date(year =
int(enddates[0]), month = int(enddates[1]), day =
int(enddates[2]))
        barn_disease_results = {}
        barn_disease_results["barns"] =
Barn.objects.all().order_by("barnname")
        if request.user.is_authenticated:
            user_household =
Household.objects.select_related("locationplace",

"locationplace__cattlecoutry",

"locationplace__cattlecoutry__cregion",

"users").\
                filter(users__in = [request.user])
            user_household = user_household[0]
            barn_disease_results["household"] = user_household
            if barn_id == -1:
                barn_cattles =
Cattle.objects.filter(barnlocation__household=user_household,

state__in=["A", "D"]). \
                    order_by("-fullbirthdate")
            else:
                barn_cattles =
Cattle.objects.filter(barnlocation__household=user_household,

barnlocation__id=barn_id,

state__in=["A", "D"]). \
                    order_by("-fullbirthdate")
                barn_disease_results["barn"] =
Barn.objects.get(id=barn_id)
            barn_disease_results["cattles"] = barn_cattles
            if not startdate and not enddate:
                cnow = datetime.datetime.now()
                cmonth = cnow.month
                cyear = cnow.year
                firstdate = datetime.date(day=1, month=cmonth,
year=cyear)
                disease_dynamics = []
                if cmonth > 11:
                    lastdate = datetime.date(day=1, month=1,
year=cyear+1) - datetime.timedelta(days = 1)
            elif not enddate and startdate:
                cmonth = startdate.month
                cyear = startdate.year
                firstdate = datetime.date(day=1, month=cmonth,
year=cyear)
                disease_dynamics = []
                if cmonth > 11:
```

```
                    lastdate = datetime.date(day=1, month=1,
year=cyear + 1) - datetime.timedelta(days=1)
                firstdate = firstdate +
relativedelta(months=12)
                lastdate = lastdate + relativedelta(months=12)
            elif not startdate and enddate:
                cmonth = enddate.month
                cyear = enddate.year
                firstdate = datetime.date(day=1, month=cmonth,
year=cyear)
                disease_dynamics = []
                if cmonth > 11:
                    lastdate = datetime.date(day=1, month=1,
year=cyear + 1) - datetime.timedelta(days=1)
                firstdate = firstdate -
relativedelta(months=12)
                lastdate = lastdate - relativedelta(months=12)
            else:
                lastdate = enddate
                firstdate = startdate
            diseases =
Diseasetype.objects.all().order_by("diseasename")
            barn_disease_results["diseases"] = diseases
            fd = firstdate - relativedelta(months=11)
            ld = lastdate - relativedelta(months=11)
            disease_dynamics2 = []
            for disease in diseases:
                firstdate = fd
                lastdate = ld
                dd = []
                dd2 = []
                for ind in range(12):
                    disease_set_count =
Cattledisease.objects.filter(Q(diseasestart__gte=firstdate,

diseaseend__gte = lastdate,

diseasestart__lte = lastdate) | \

Q(diseasestart__gte=firstdate,

diseaseend__lte=lastdate) | \

Q(diseasestart__lte=firstdate,

diseaseend__gte=firstdate,

diseaseend__lte=lastdate) | \

Q(diseasestart__gte=firstdate,

diseaseend__lte=lastdate),
```

```
                                                                cattle__in
=barn_cattles,

disease = disease).\
                            count()
                        if barn_id != -1:
                            disease_set_count_ =
Cattledisease.objects.filter(Q(diseasestart__gte=firstdate,

diseaseend__gte=lastdate,

diseasestart__lte=lastdate) | \

Q(diseasestart__gte=firstdate,

diseaseend__lte=lastdate) | \

Q(diseasestart__lte=firstdate,

diseaseend__gte=firstdate,

diseaseend__lte=lastdate) | \

Q(diseasestart__gte=firstdate,

diseaseend__lte=lastdate),

cattle__barnlocation__household=user_household,

disease=disease). \
                                  count()

dd.append([str(firstdate.month)+'.'+str(firstdate.year),

disease_set_count])

dd2.append([str(firstdate.month)+'.'+str(firstdate.year),

disease_set_count_])
                        else:

dd.append((str(firstdate.month)+'.'+str(firstdate.year),

disease_set_count))
                        firstdate = firstdate +
relativedelta(months=1)
                        lastdate = lastdate +
relativedelta(months=1)
                    disease_dynamics.append(dd)
                    disease_dynamics2.append(dd2)
                barn_disease_results["dynamics"] = disease_dynamics
                barn_disease_results["dynamicsgeneral"] =
disease_dynamics2
```

```
          return render(request, "barndiseasedynamics.html",
barn_disease_results)

      def breed_disease_dynamics_view(request, cattle_breed_id =
-1, startdate = None, enddate = None):
          if startdate:
              startdates = startdate.split('-')
              startdate = datetime.datetime.date(year =
int(startdates[0]), month = int(startdates[1]), day =
int(startdates[2]))
          if enddate:
              enddates = enddate.split('-')
              enddate = datetime.datetime.date(year =
int(enddates[0]), month = int(enddates[1]), day =
int(enddates[2]))
          breed_disease_results = {}
          breed_disease_results["breeds"] =
Cattlebreed.objects.all().order_by("cbreedname")
          if request.user.is_authenticated:
              user_household =
Household.objects.select_related("locationplace",

"locationplace__cattlecoutry",

"locationplace__cattlecoutry__cregion",

"users").\
                  filter(users__in = [request.user])
              user_household = user_household[0]
              breed_disease_results["household"] = user_household
              if cattle_breed_id == -1:
                  breed_cattles =
Cattle.objects.filter(barnlocation__household=user_household,

state__in=["A", "D"]). \
                       order_by("-fullbirthdate")
              else:
                  breed_cattles =
Cattle.objects.filter(barnlocation__household=user_household,

cattlebreed__id=cattle_breed_id,

state__in=["A", "D"]). \
                       order_by("-fullbirthdate")
                  breed_disease_results["breed"] =
Cattlebreed.objects.get(id=cattle_breed_id)
              breed_disease_results["cattles"] = breed_cattles
              if not startdate and not enddate:
                  cnow = datetime.datetime.now()
                  cmonth = cnow.month
                  cyear = cnow.year
                  firstdate = datetime.date(day=1, month=cmonth,
year=cyear)
```

```
                disease_dynamics = []
                if cmonth > 11:
                    lastdate = datetime.date(day=1, month=1,
year=cyear + 1) - datetime.timedelta(days=1)
            elif not enddate and startdate:
                cmonth = startdate.month
                cyear = startdate.year
                firstdate = datetime.date(day=1, month=cmonth,
year=cyear)
                disease_dynamics = []
                if cmonth > 11:
                    lastdate = datetime.date(day=1, month=1,
year=cyear + 1) - datetime.timedelta(days=1)
                firstdate = firstdate +
relativedelta(months=12)
                lastdate = lastdate + relativedelta(months=12)
            elif not startdate and enddate:
                cmonth = enddate.month
                cyear = enddate.year
                firstdate = datetime.date(day=1, month=cmonth,
year=cyear)
                disease_dynamics = []
                if cmonth > 11:
                    lastdate = datetime.date(day=1, month=1,
year=cyear + 1) - datetime.timedelta(days=1)
                firstdate = firstdate -
relativedelta(months=12)
                lastdate = lastdate - relativedelta(months=12)
            else:
                lastdate = enddate
                firstdate = startdate
            fd = firstdate - relativedelta(months=11)
            ld = lastdate - relativedelta(months=11)
            disease_dynamics2 = []
            diseases =
Diseasetype.objects.all().order_by("diseasename")
            breed_disease_results["diseases"] = diseases
            for disease in diseases:
                firstdate = fd
                lastdate = ld
                dd = []
                dd2 = []
                for ind in range(12):
                    disease_set_count =
Cattledisease.objects.filter(Q(diseasestart__gte=firstdate,

diseaseend__gte = lastdate,

diseasestart__lte = lastdate) | \

Q(diseasestart__gte=firstdate,

diseaseend__lte=lastdate) | \
```

```
                                                    Q(diseasestart__lte
=firstdate,

diseaseend__gte=firstdate,

diseaseend__lte=lastdate) | \

Q(diseasestart__gte=firstdate,

diseaseend__lte=lastdate),

cattle__in=breed_cattles,

disease = disease).\
                            count()
                        if cattle_breed_id != -1:
                            disease_set_count_ =
Cattledisease.objects.filter(Q(diseasestart__gte=firstdate,

diseaseend__gte=lastdate,

diseasestart__lte=lastdate) | \

Q(diseasestart__gte=firstdate,

diseaseend__lte=lastdate) | \

Q(diseasestart__lte=firstdate,

diseaseend__gte=firstdate,

diseaseend__lte=lastdate) | \

Q(diseasestart__gte=firstdate,

diseaseend__lte=lastdate),

cattle__barnlocation__household=user_household,

disease=disease). \
                                count()
                            dd.append([str(firstdate.month) + '.' +
str(firstdate.year),
                                       disease_set_count])
                            dd2.append([str(firstdate.month) + '.'
+ str(firstdate.year),
                                        disease_set_count_])
                        else:
                            dd.append((str(firstdate.month) + '.' +
str(firstdate.year),
                                       disease_set_count))
                        firstdate = firstdate +
relativedelta(months=1)
```

```
                    lastdate = lastdate +
relativedelta(months=1)
                disease_dynamics.append(dd)
                disease_dynamics2.append(dd2)
            breed_disease_results["dynamics"] =
disease_dynamics
            breed_disease_results["generaldynamics"] =
disease_dynamics2
        return render(request, "breeddiseasedynamics.html",
breed_disease_results)
```

ДОДАТОК В

Слайди презентації

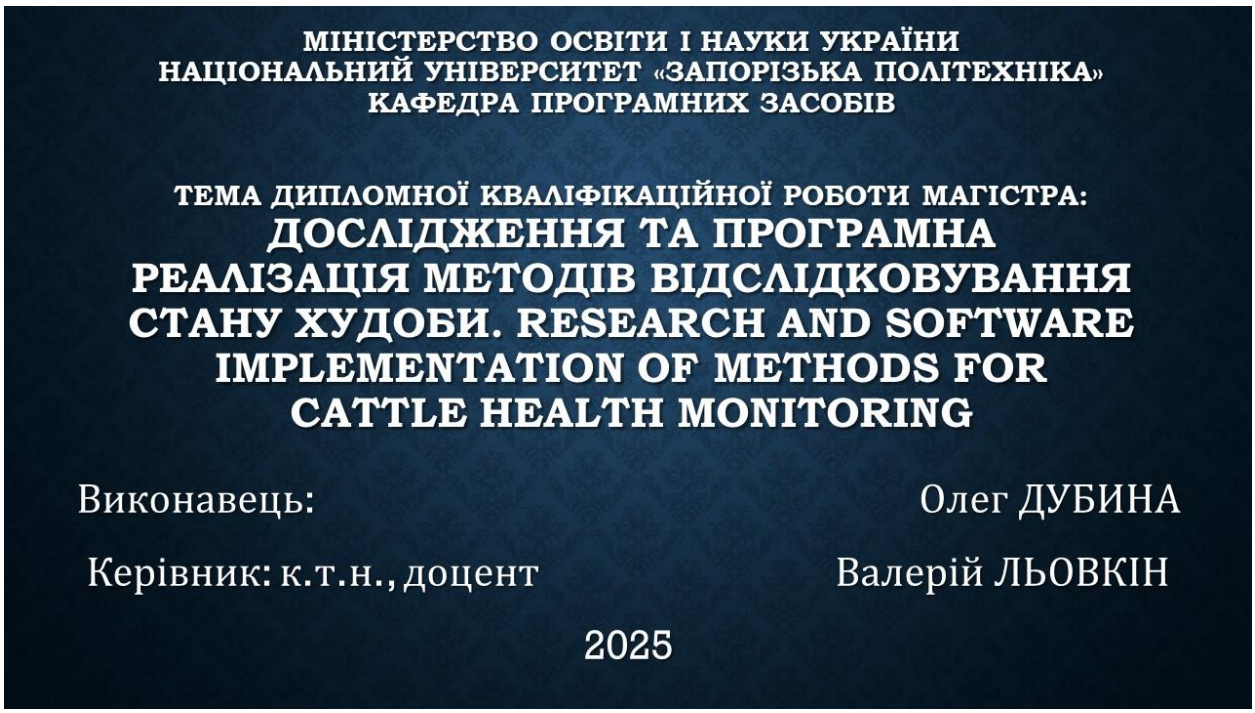

Рисунок В.1 – Початковий слайд

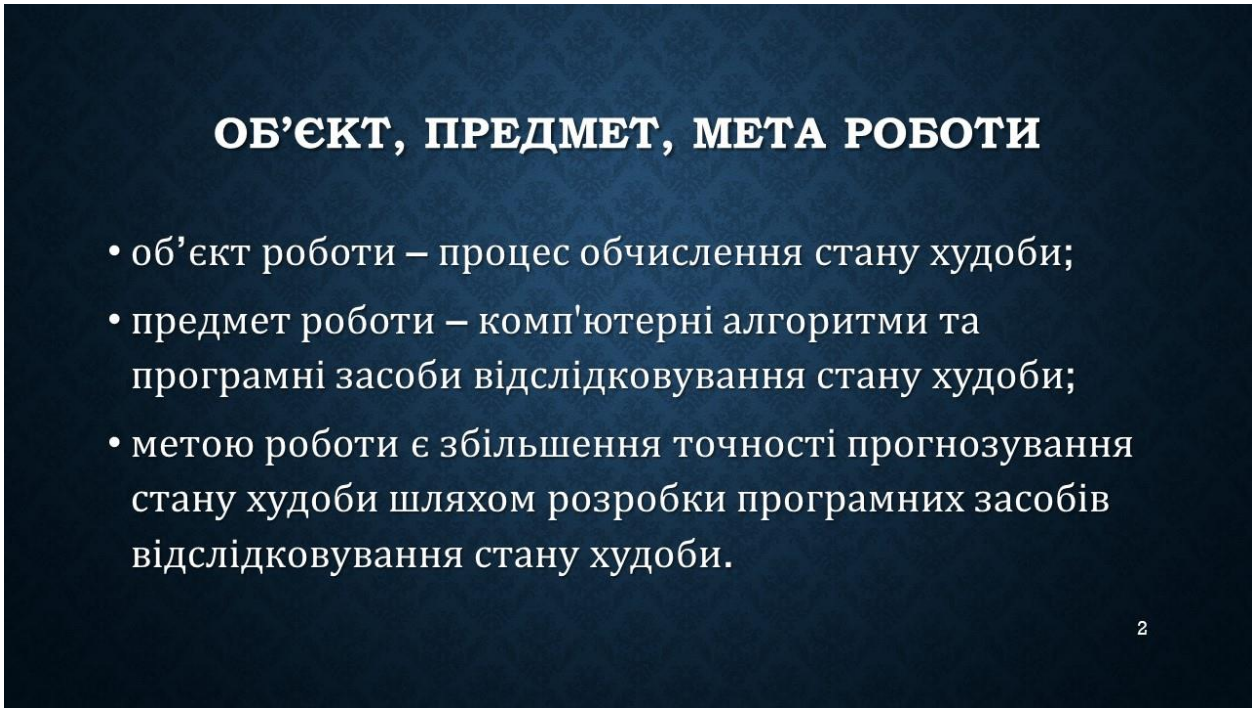

Рисунок В.2 – Об'єкт, предмет, мета роботи

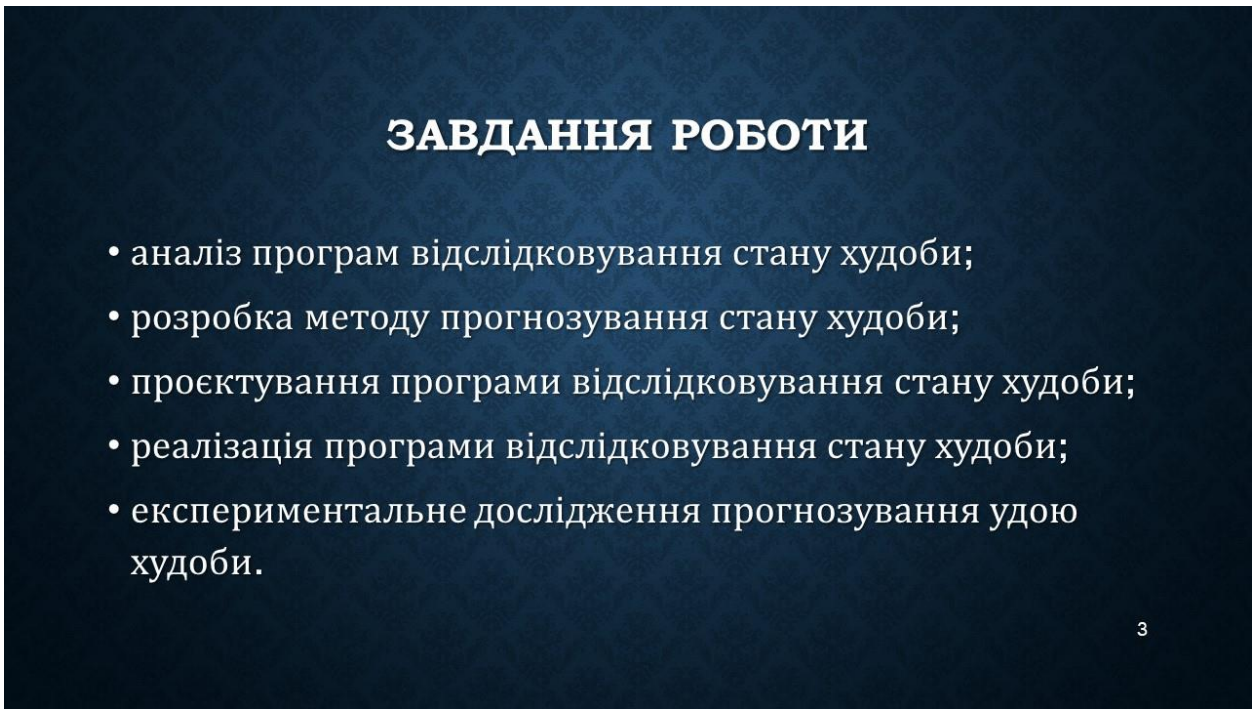

Рисунок В.3 – Завдання роботи

Рисунок В.4 – Порівняння аналогів вебзастосунку відслідковування стану худоби

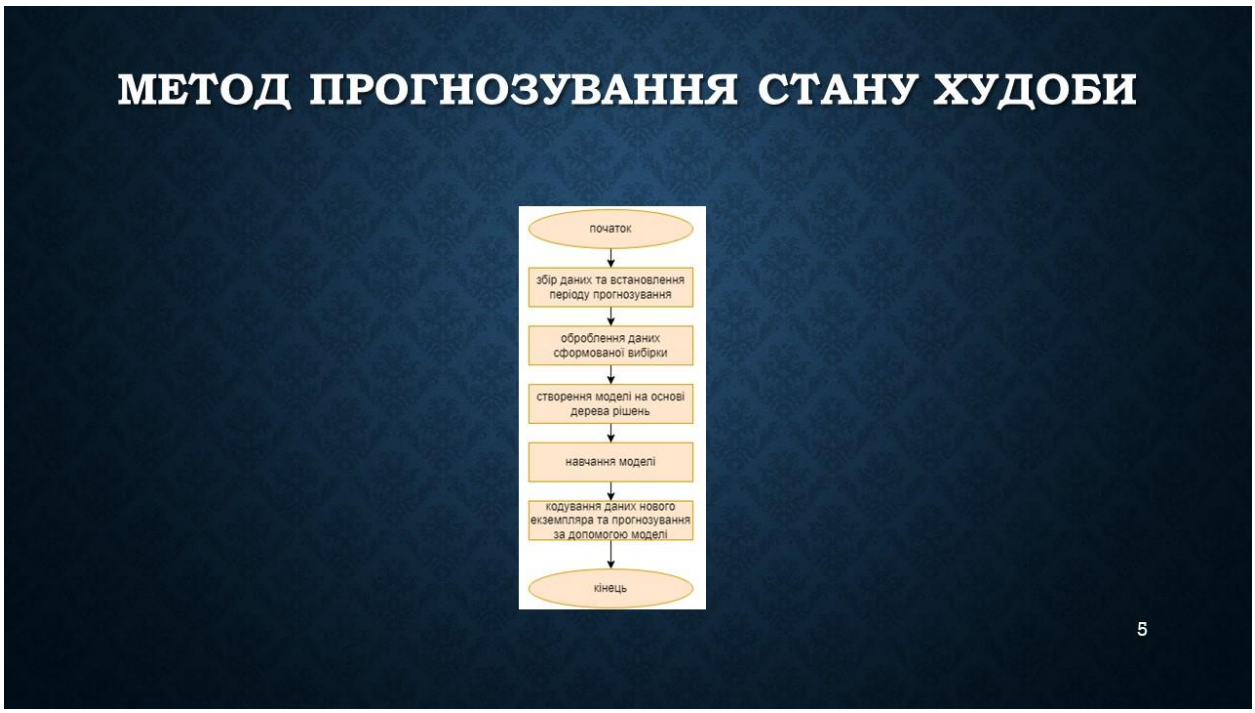

Рисунок В.5 – Метод прогнозування стану худоби

Рисунок В.6 – Вибір мови для реалізації програми відслідковування стану худоби

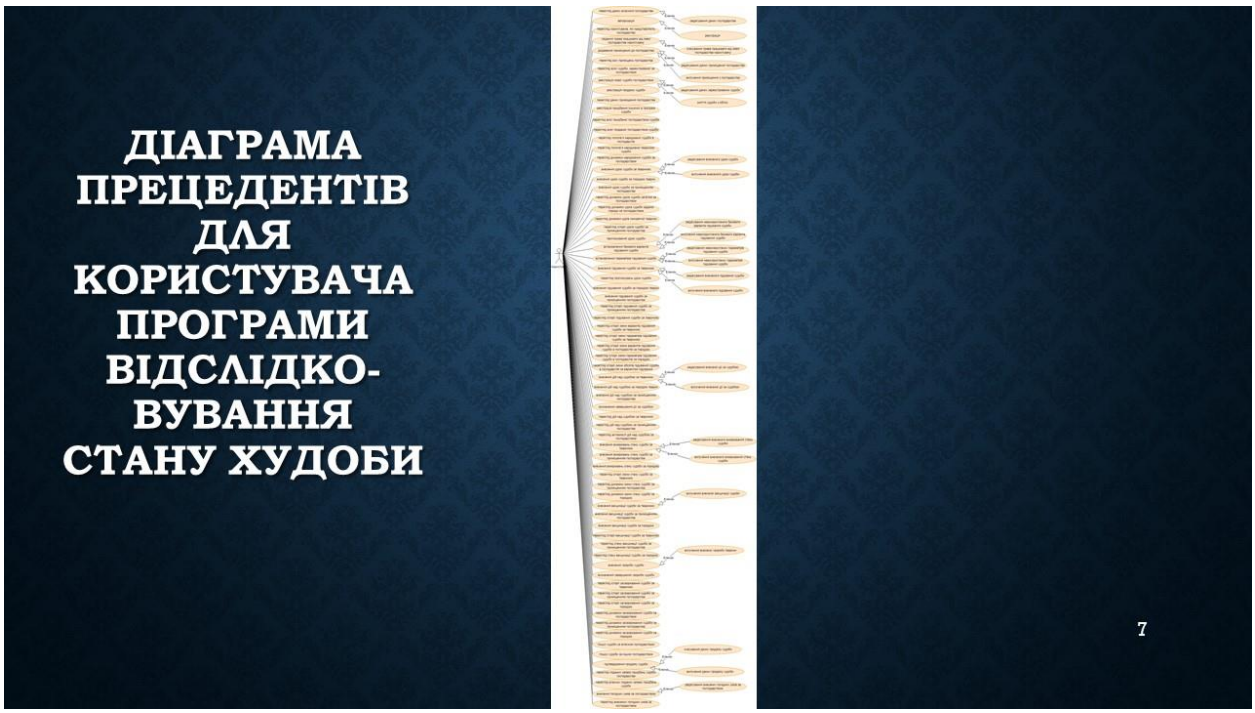

Рисунок В.7 – Діаграма прецедентів для користувача програми відслідковування стану худоби

Рисунок В.8 – Схема бази даних програми відслідковування стану худоби

Рисунок В.9 – Функціональна схема програми відслідковування стану худоби

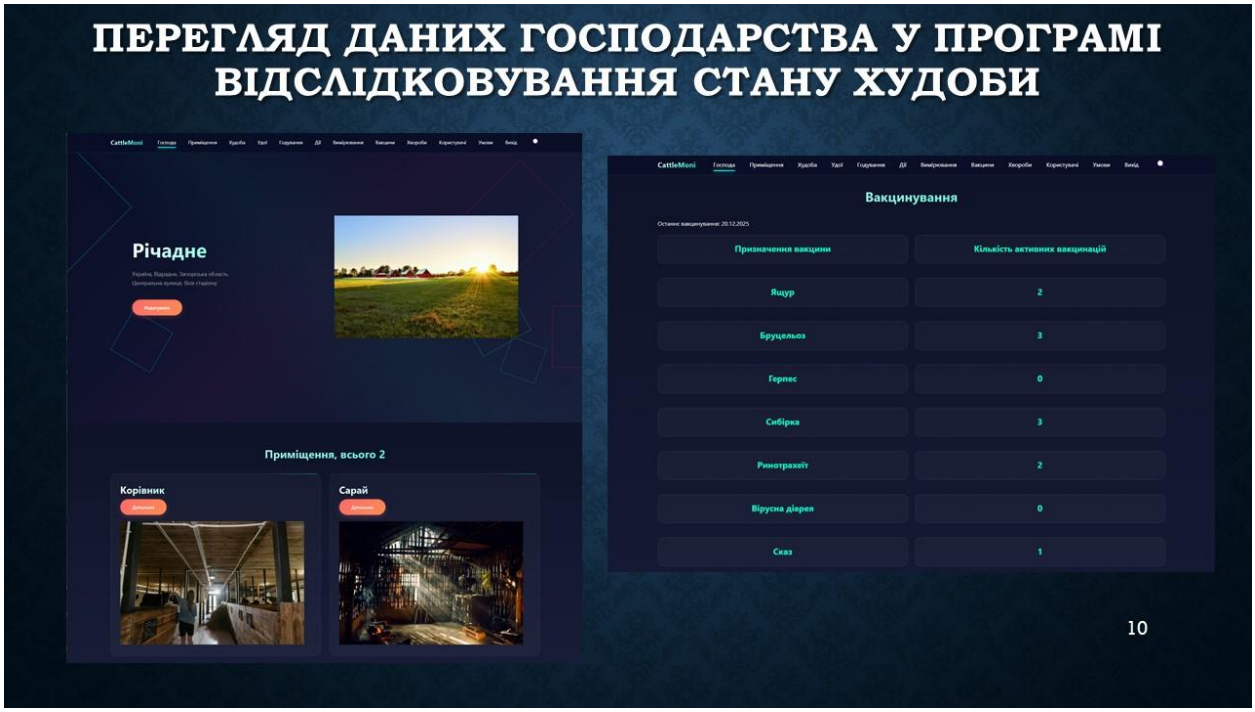

Рисунок В.10 – Перегляд даних господарства у програмі відслідковування стану худоби

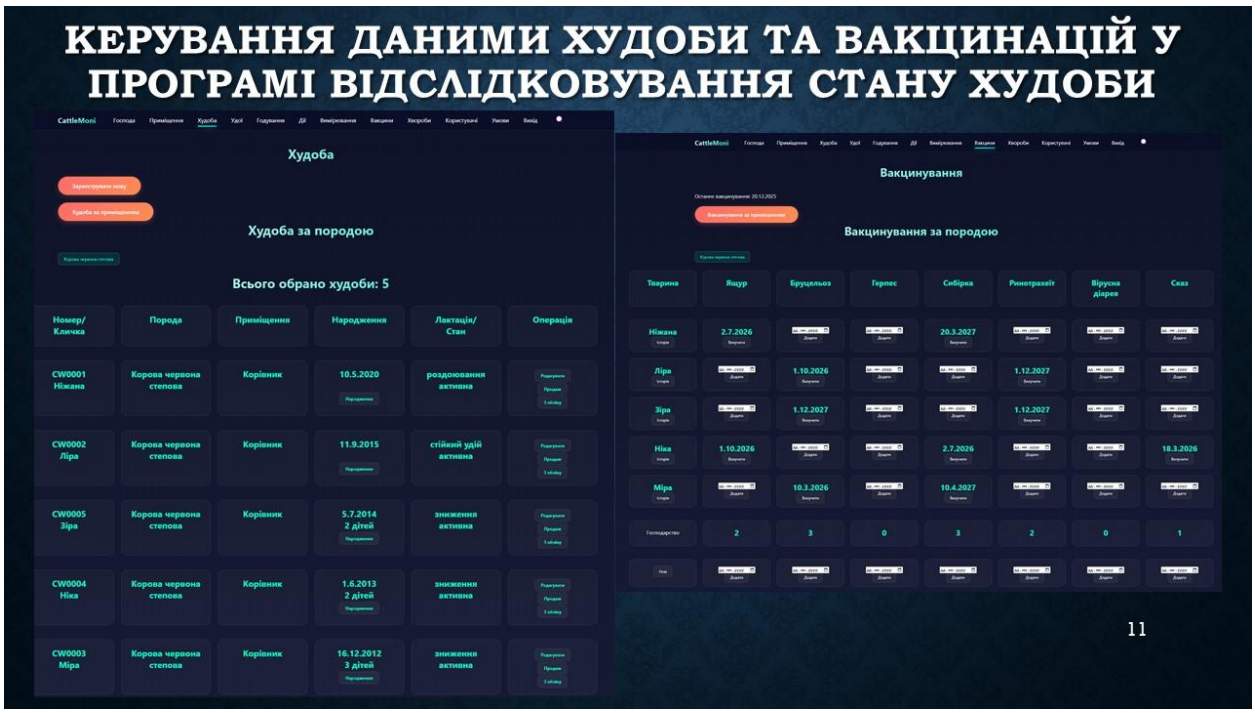

Рисунок В.11 – Керування даними худоби та вакцинацій у програмі відслідковування стану худоби

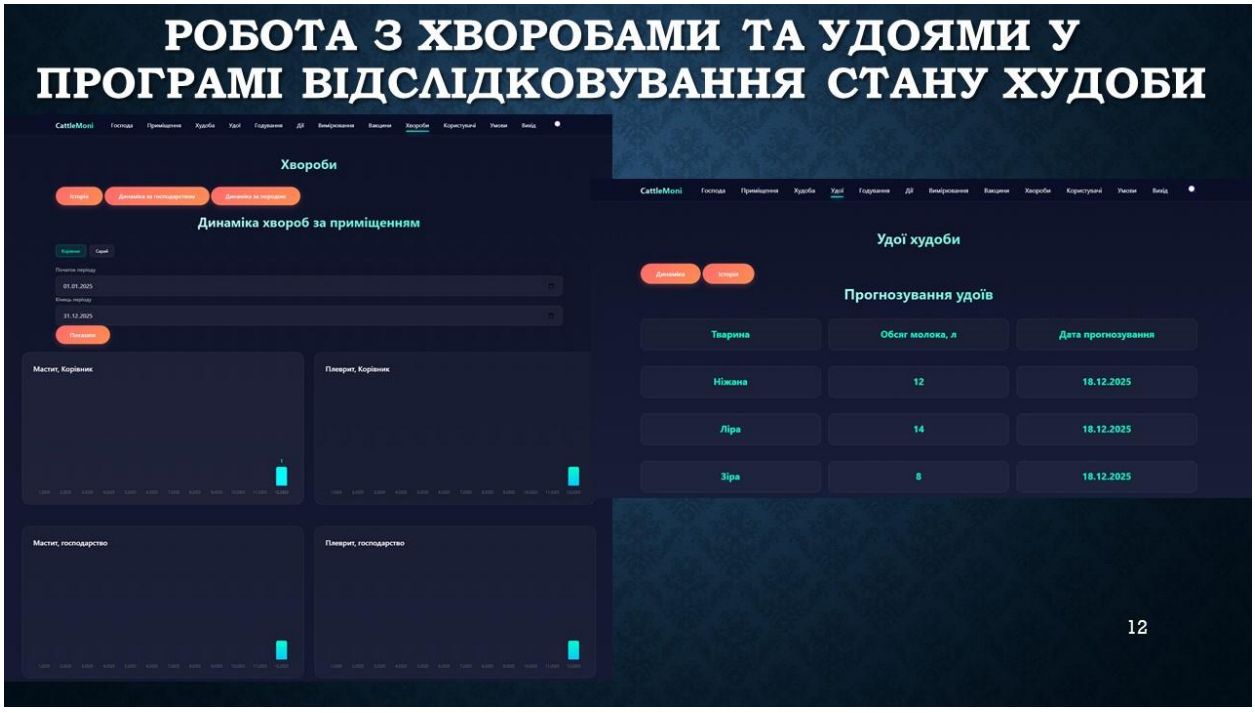

Рисунок В.12 – Робота з хворобами та удоями у програмі відслідковування стану худоби

РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТАЛЬНОГО ДОСЛІДЖЕННЯ ПРОГНОЗУВАННЯ УДОЮ ХУДОБИ

Метод / модель	Середня абсолютна похибка	Кореневе середньо-квадратичне відхилення	Середньо-квадратична похибка
Метод опорних векторів	0,0019092259	0,0021246919	0,0000045143
Дерево рішень (запропонований метод)	0,0003046144	0,0003837075	0,0000001472

13

Рисунок В.13 – Результати експериментального дослідження прогнозування удою худоби

ВИСНОВКИ

- У роботі повністю досягнуто мету, що передбачала збільшення точності прогнозування стану худоби шляхом розробки програмних засобів відслідковування стану худоби. Збільшення точності прогнозування стану худоби забезпечено створенням і навчанням моделі прогнозування на основі запропонованого методу, програмною реалізацією такої моделі та її застосуванням у відповідних сценаріях для роботи з користувачем на основі зібраних даних про стан худоби.
- Наукова новизна роботи полягає в методі прогнозування стану худоби, що характеризується використанням дерев рішень, визначенням обчислення вхідних ознак, що дозволяє визначати обсяг удоїв худоби на наступний період.
- Проведено експериментальне дослідження прогнозування обсягу удоїв молока худоби. Запропонований метод дозволив зменшити рівень похибки прогнозування стану худоби у вигляді обсягу удоїв молока порівняно з альтернативою у вигляді методу опорних векторів.

14

Рисунок В.14 – Висновки