

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 проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ОРЕНДИ
СІЛЬСЬКОГОСПОДАРСЬКИХ ТВАРИН
FARM ANIMAL RENTAL SOFTWARE

Виконав(ла): студент(ка) 4 курсу, групи КНТ-211

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

ВІННІЧЕНКО Д.В.

(ПРИЗВИЩЕ та ініціали)

Керівник СТЕПАНЕНКО О.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ЗЕЛІК О.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 122 Комп'ютерні науки
 (КОД І НАЙМЕНУВАННЯ)
 Освітня програма (спеціалізація) Комп'ютерні науки
 (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)

ВІННІЧЕНКА Дмитра Вадимовича
 (ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення оренди
сільськогосподарських тварин. Farm Animal Rental Software

керівник проєкту (роботи) к.т.н., доцент СТЕПАНЕНКО Олександр Олексійович,
 (науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 02 червня 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. Проєктування програми. 3. Розробка програми. 4. Експлуатація та тестування програми.

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

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СТЕПАНЕНКО О.О., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання «14» квітня 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.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Дмитро ВІННІЧЕНКО
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Олександр СТЕПАНЕНКО
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 91 с., 2 табл., 26 рис., 3 дод., 8 джерел.

ТВАРИНА, ОРЕНДА, ФЕРМЕР, ЗАМОВЛЕННЯ, ВЕБЗАСТОСУНОК.

Об'єкт роботи – процеси оренди сільськогосподарських тварин.

Предмет роботи – програмне забезпечення оренди сільськогосподарських тварин.

Мета роботи – розробити програмне забезпечення оренди сільськогосподарських тварин для надання можливості користувачам знаходити варіанти тимчасового володіння тваринами в господарських завданнях.

Матеріали, методи та технічні засоби: мова Python і фреймворк Django, система керування базою даних MySQL.

Результати. Створена програма призначена для оренди сільськогосподарських тварин у власників на певний період часу, надання власниками своїх сільськогосподарських тварин для оренди. Реалізовані програмні одиниці застосунку Django для забезпечення всіх функцій програми за структурою шаблону Model-View-Template.

Висновки. Програма дозволяє фермерам виставляти свої пропозиції оренди сільськогосподарських тварин, орендарям – шукати підходящу для них пропозицію, замовляти сільськогосподарських тварин для оренди, керувати цими замовленнями, залишати відгуки про оренду, надавати дані про збір продуктів з сільськогосподарських тварин.

Галузь використання. Оренда сільськогосподарських тварин на період від тижня до року та більше.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 91 pages, 2 tables, 26 figures, 3 appendixes, 8 sources.

ANIMAL, RENT, FARMER, ORDER, WEB APPLICATION.

The object of the work is the processes of renting farm animals.

The subject of the work is the software for renting farm animals.

The purpose of the work is to develop software for renting farm animals to enable users to find options for temporary ownership of animals in economic tasks.

Materials, methods and technical tools: Python language and Django framework, MySQL database management system.

Results. The created program is designed to rent farm animals from owners for a certain period of time, and for owners to provide their farm animals for rent. The Django application software units are implemented to provide all the functions of the program according to the Model-View-Template structure.

Conclusions. The program allows farmers to list their offers for renting farm animals, tenants to search for a suitable offer for them, order farm animals for rent, manage these orders, leave reviews about the rental, provide data on the collection of products from farm animals.

Field of use. Renting farm animals for a period from a week to a year and more.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз проблеми та постановка завдань дослідження	9
1.1 Стан розробки програм оренди сільськогосподарських тварин	9
1.2 Порівняння програм оренди сільськогосподарських тварин	12
1.3 Висновки за розділом 1	13
2 Проєктування програми.....	14
2.1 Обґрунтування вибору засобів розробки програми	14
2.2 Визначення вимог користувачів до програми оренди сільськогосподарських тварин	15
2.3 Проєктування структури бази даних.....	22
2.4 Висновки за розділом 2	32
3 Розробка програми	33
3.1 Порядок роботи над функціями програми	33
3.2 Реалізація програми оренди сільськогосподарських тварин.....	34
3.3 Висновки за розділом 3	48
4 Експлуатація та тестування програми	49
4.1 Призначення програми	49
4.2 Опис роботи з програмою оренди сільськогосподарських тварин та тестування.....	50
4.3 Висновки за розділом 4	54
Висновки	55
Перелік джерел посилання	57
Додаток А Технічне завдання	58
Додаток Б Текст програми	63
Додаток В Слайди презентації.....	85

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

MVT – Model-View-Template;

ПЗ – програмне забезпечення.

ВСТУП

Володіння сільськогосподарськими тваринами є тривалим і відповідальним процесом, однак не всі люди готові стати учасниками цього процесу. З іншого боку оренда дозволяє спробувати будь-якому бажаючому те, на скільки він готовий зануритись у такий процес. Окрім того під час оренди можна збирати продукти, таким чином отримуючи певний економічний зиск. Наприклад, корови дозволяють отримати молоко, сир, курки – яйця. Усе це за певних умов, а особливо в складних економічних умовах, може бути економічно вигідно. Відповідно людина може на певний період часу стати учасником процесу володіння сільськогосподарськими тваринами, а потім це змінити, коли умови її життя змінились. При цьому тварина без проблем для її існування може повернутись до власника. Це змінює можливості людей, дозволяє взаємодіяти їм з природою у обмежений період часу. Застосування програмного забезпечення (ПЗ) для реалізації таких задач є необхідною умовою в сучасному світі. У той же час ПЗ, яке б повноцінно дозволяло одним користувачам створювати свої пропозиції оренди сільськогосподарських тварин, а іншим – знаходити і використовувати такі пропозиції, відсутнє.

У подальшій роботі необхідно виконати мету – розробити ПЗ оренди сільськогосподарських тварин для надання можливості користувачам знаходити варіанти тимчасового володіння тваринами в господарських завданнях.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Стан розробки програм оренди сільськогосподарських тварин

Існуючі рішення для оренди сільськогосподарських тварин доволі обмежено реалізують дану ідею, хоча варто вказати на те, що їх наявність додатково свідчить про корисність ідеї такої оренди.

Програма Rentthechicken [1] (рис. 1.1) отримала популярність у Сполучених Штатах Америки у зв'язку з кризою через підвищену вартість курячих яєць, що реалізувалась цього року.

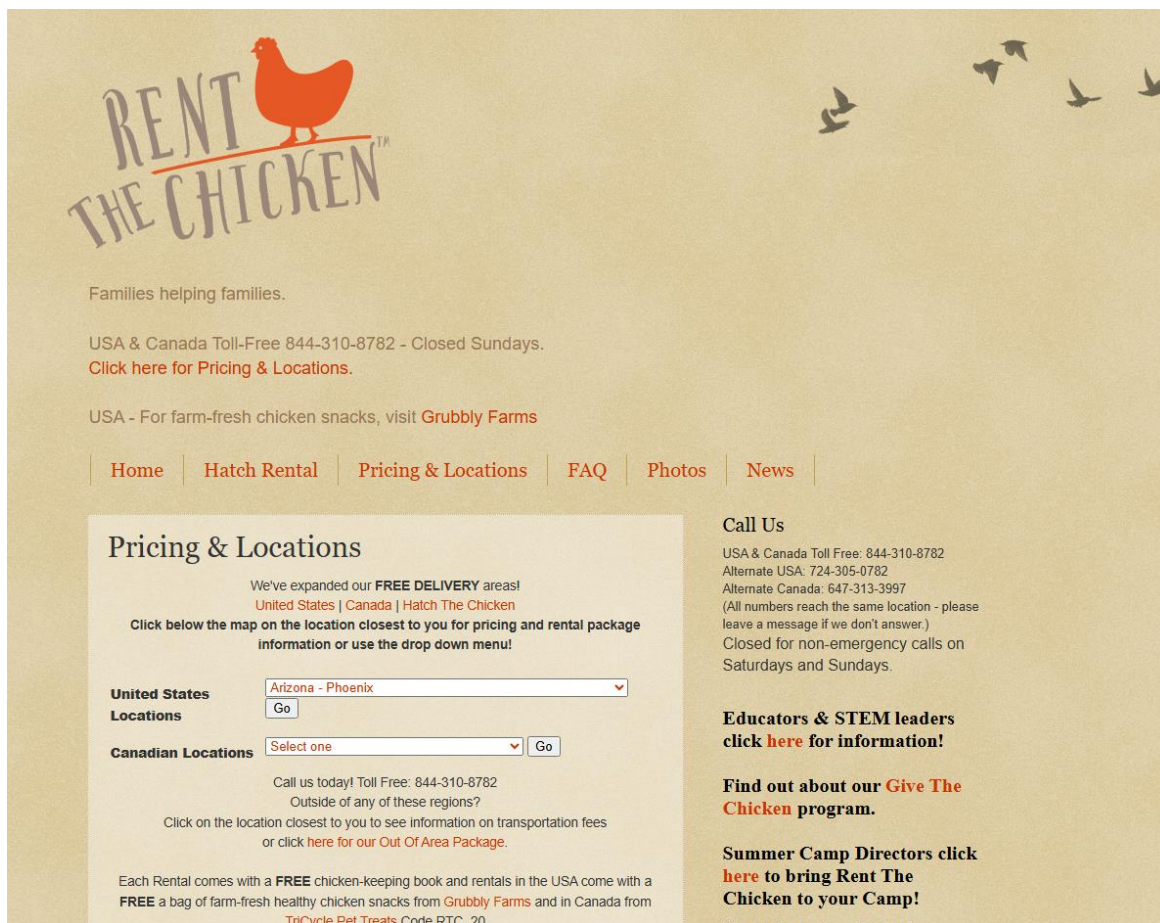
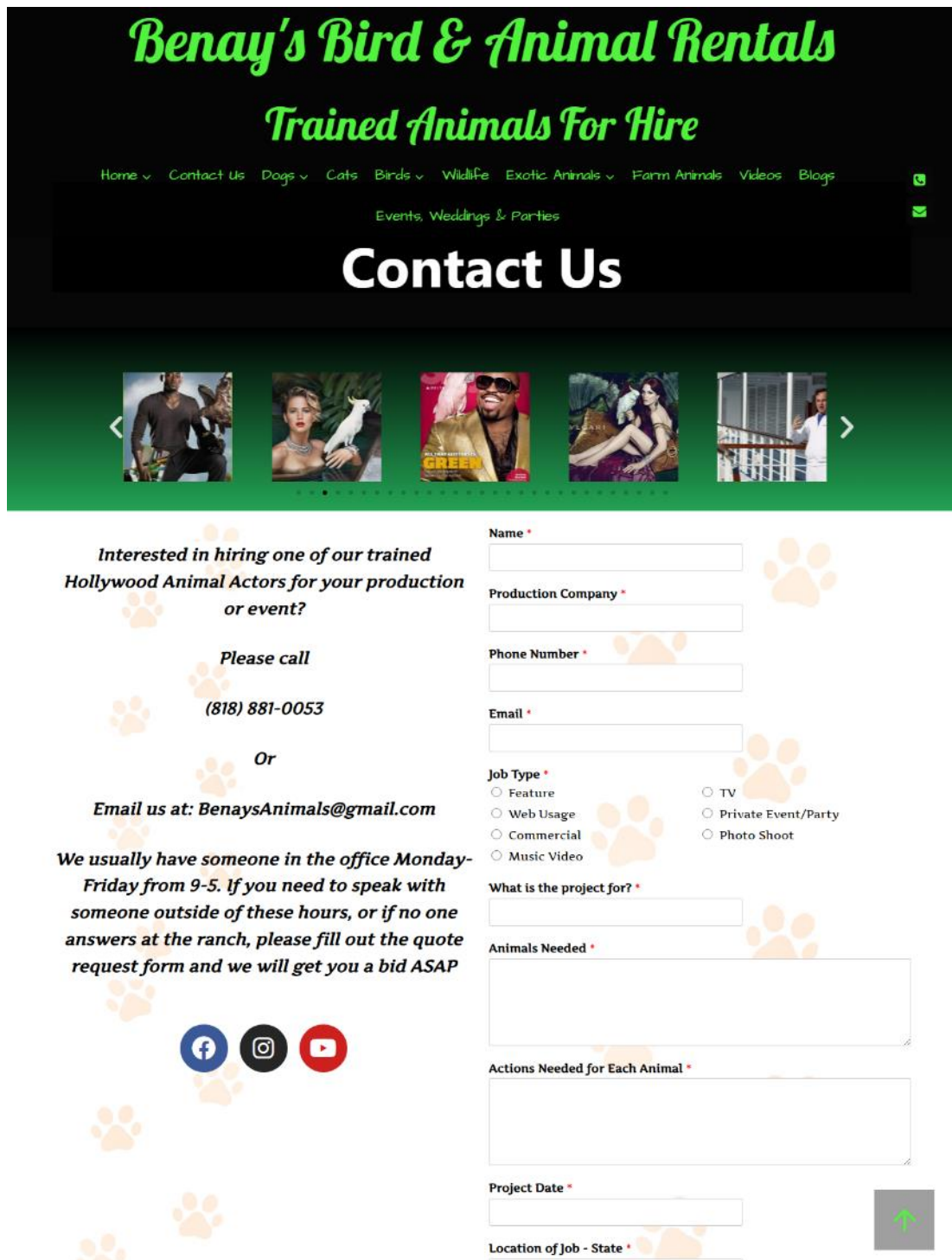


Рисунок 1.1 – Вебінтерфейс Rentthechicken

Програма Rentthechicken більшою мірою має інформаційні функції, просуваючи ідею оренди сільськогосподарських тварин. У програмі можна за регіоном у Сполучених Штатах Америки, Канаді побачити вартість та умови

доставки курей, також надається форма для контактування, де потрібно задати ім'я, адресу, номер телефону.

Benaysanimals [2] (рис. 1.2) дозволяє орендувати сільськогосподарських тварин для проведення певних подій (кіно- та фотозйомок).



Benay's Bird & Animal Rentals
Trained Animals For Hire

Home ▾ Contact Us Dogs ▾ Cats Birds ▾ Wildlife Exotic Animals ▾ Farm Animals Videos Blogs

Events, Weddings & Parties

Contact Us

Interested in hiring one of our trained Hollywood Animal Actors for your production or event?

Please call
 (818) 881-0053

Or

Email us at: BenaysAnimals@gmail.com

We usually have someone in the office Monday-Friday from 9-5. If you need to speak with someone outside of these hours, or if no one answers at the ranch, please fill out the quote request form and we will get you a bid ASAP

Facebook Instagram YouTube

Name *

Production Company *

Phone Number *

Email *

Job Type *

☐ Feature ☐ TV

☐ Web Usage ☐ Private Event/Party

☐ Commercial ☐ Photo Shoot

☐ Music Video

What is the project for? *

Animals Needed *

Actions Needed for Each Animal *

Project Date *

Location of Job - State *

Рисунок 1.2 – Вебінтерфейс Benaysanimals

Програма Benaysanimals надає можливість орендувати також і інших тварин, зокрема екзотичних. Уся функціональність полягає в інформування та заповненні форми для контактування.

Така ж ідея контактування на основі форми представлена і в застосунку A-Z Animals [3].

Програма RentMotherNature [4] (рис. 1.3) направлена не на оренду сільськогосподарських тварин фактично, а на отримання тих продуктів, які від цих тварин можна отримати. Наприклад, козячого сиру, таким чином орендуючи результати роботи тварини. Відповідно таку сільськогосподарську тварину не потрібно доставляти до себе, а такий варіант оренди є коротким (порції на день відповідно).

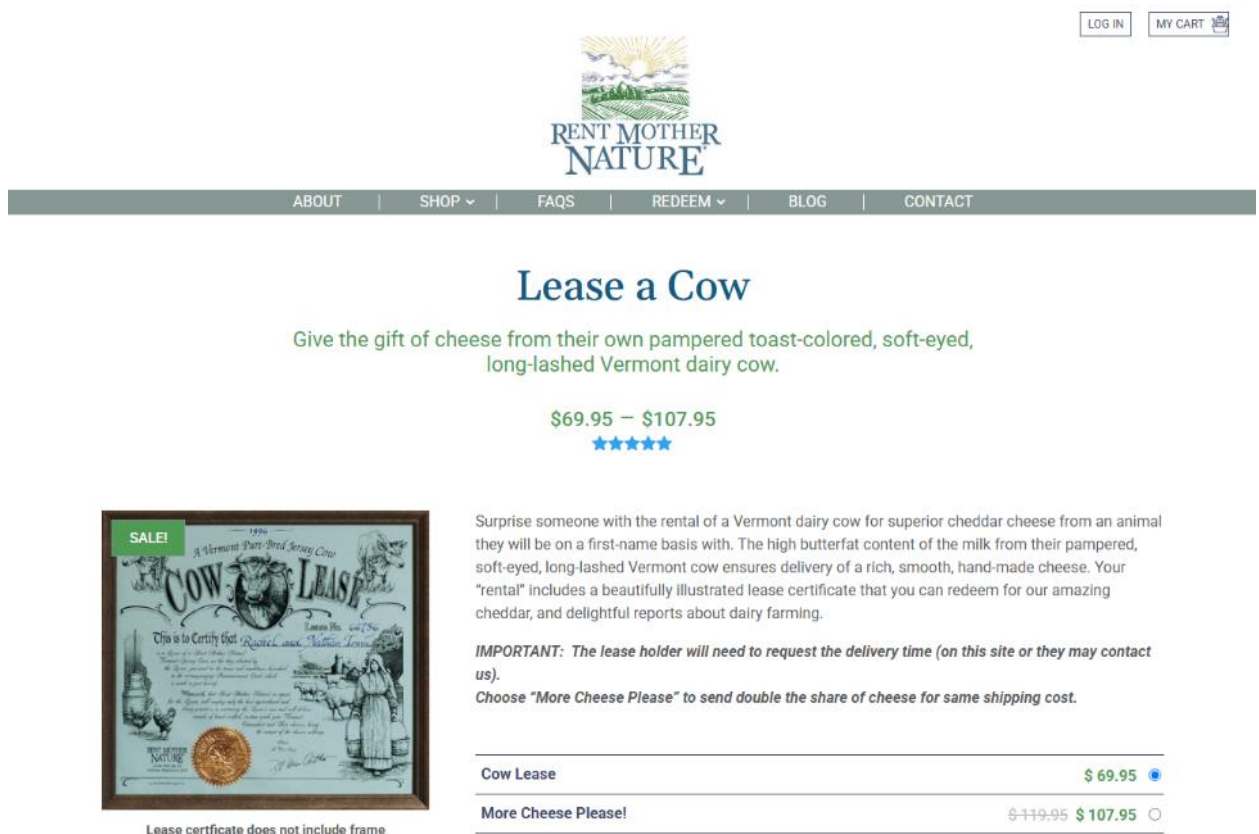


Рисунок 1.3 – Вебінтерфейс RentMotherNature

Така ж ідея отримала розвиток і в Швейцарії, де на туристичному вебсайті міста Аппенцелль [5] представляється послуга оренди корів (рис. 1.4).

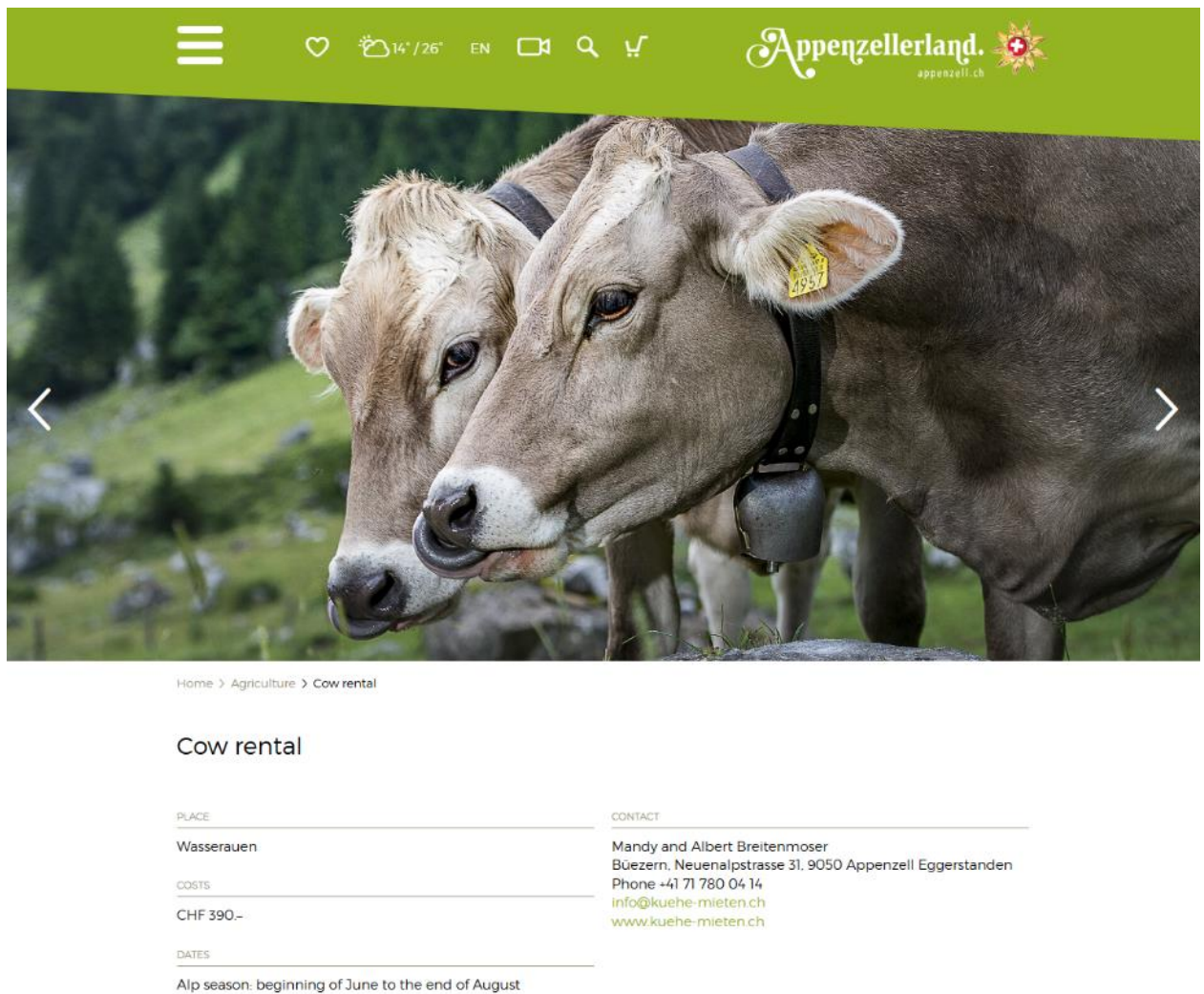


Рисунок 1.4 – Вебсайт appenzell.ch

1.2 Порівняння програм оренди сільськогосподарських тварин

Підсумкове порівняння програм оренди сільськогосподарських тварин включає Rentthechicken, Benaysanimals і власну розробку, названу Farmrenter (табл. 1.1). Власна розробка направлена на те, щоб інтегрувати функції традиційного замовлення оренди до оренди сільськогосподарських тварин, що відсутні у аналогів. Аналоги тільки надають форми для залишення контакту, щоб потім зв'язатися з цими замовниками. Відповідно як таких замовлень оренди сільськогосподарських тварин взагалі немає.

Таблиця 1.1 – Порівняння програм оренди сільськогосподарських тварин

Характеристика	Rentthechicken	Benaysanimals	Farmrenter
Оформлення замовлення на оренду сільсько-господарських тварин	Без конкретних дат	Без конкретних дат	З датами та кількістю
Множинне представлення фермерів-власників тварин	Немає	Немає	Надається
Визначення обсягів отриманих продуктів від оренди	Немає	Немає	Надається
Керування власними замовленнями оренди сільськогосподарських тварин	Немає	Немає	Надається

1.3 Висновки за розділом 1

Встановлені завдання роботи:

- аналіз стану проблеми розробки програм оренди сільськогосподарських тварин;
- проєктування програми оренди сільськогосподарських тварин;
- реалізація, тестування програми оренди сільськогосподарських тварин.

Стан проблеми розробки програм оренди сільськогосподарських тварин розглянуто в цьому розділі. Існуючі програми представляють ідею оренди сільськогосподарських тварин, але функціонально містять тільки форму для подання заявки.

2 ПРОЄКТУВАННЯ ПРОГРАМИ

2.1 Обґрунтування вибору засобів розробки програми

Засоби розробки програми оренди сільськогосподарських тварин включають:

– мову програмування Python, яка є гнучкою, простою для вивчення, що важливо при використанні додаткових засобів, має зручний синтаксис, який посилює досягнення завдань, характеризується зменшенням часу розробки;

– фреймворк Django [6] для розробки ПЗ як вебзастосунку, який передбачає створення застосунку на основі структури Model-View-Template (MVT), може застосовуватись для великих, середніх проєктів, керованих даними, до яких і можна віднести даний, окрім того має вбудований рушій для створення шаблонів вебсторінок, який було порівняно з CherryPy [7] (табл. 2.1).

Таблиця 2.1 – Порівняння засобів розробки програми оренди сільськогосподарських тварин

Характеристика	CherryPy	Django
Структура застосунку	Визначається розробником	MVT
Застосування	Невеликі прості проєкти	Великі, середні проєкти, керовані даними
Об'єктно-реляційне відображення	Немає, можна додати	Одразу після встановлення готове до використання
Рушій для створення шаблонів вебсторінок	Немає, можна додати	Одразу після встановлення готовий до використання

2.2 Визначення вимог користувачів до програми оренди сільськогосподарських тварин

Оскільки надання в оренду сільськогосподарських тварин є відповідальним процесом, то фермерів-власників сільськогосподарських тварин потрібно додатково контролювати. Для цього передбачено, що адміністратори будуть створювати облікові записи таких користувачів. Відповідно їх виділено в окрему роль. Це фермери-власники. З іншого боку є фермери-орендарі, які навпаки беруть в оренду сільськогосподарських тварин, які належать фермерам-власникам. Також є неавторизовані користувачі.

Неавторизований користувач у підсумку повинен працювати з таким основним потоком сценаріїв:

- перегляд інформаційної картки сільськогосподарської тварини фермера;
- перегляд інформаційної картки породи сільськогосподарської тварини фермера;
- перегляд збору продуктів у порід сільськогосподарських тварин фермера;
- відображення сільськогосподарських тварин у фермерів для оренди;
- перегляд порід сільськогосподарських тварин у фермера;
- перегляд сільськогосподарських тварин у фермера за породою;
- перегляд порід сільськогосподарських тварин на фермі;
- перегляд сільськогосподарських тварин на фермі за породою;
- перегляд відгуків про сільськогосподарську тварину;
- перегляд відгуків про породу сільськогосподарських тварин фермера;
- перегляд відгуків про фермера;
- перегляд відгуків користувача;
- авторизація;
- перегляд інформаційної картки фермера;
- перегляд інформаційної картки ферми (рис. 2.1).

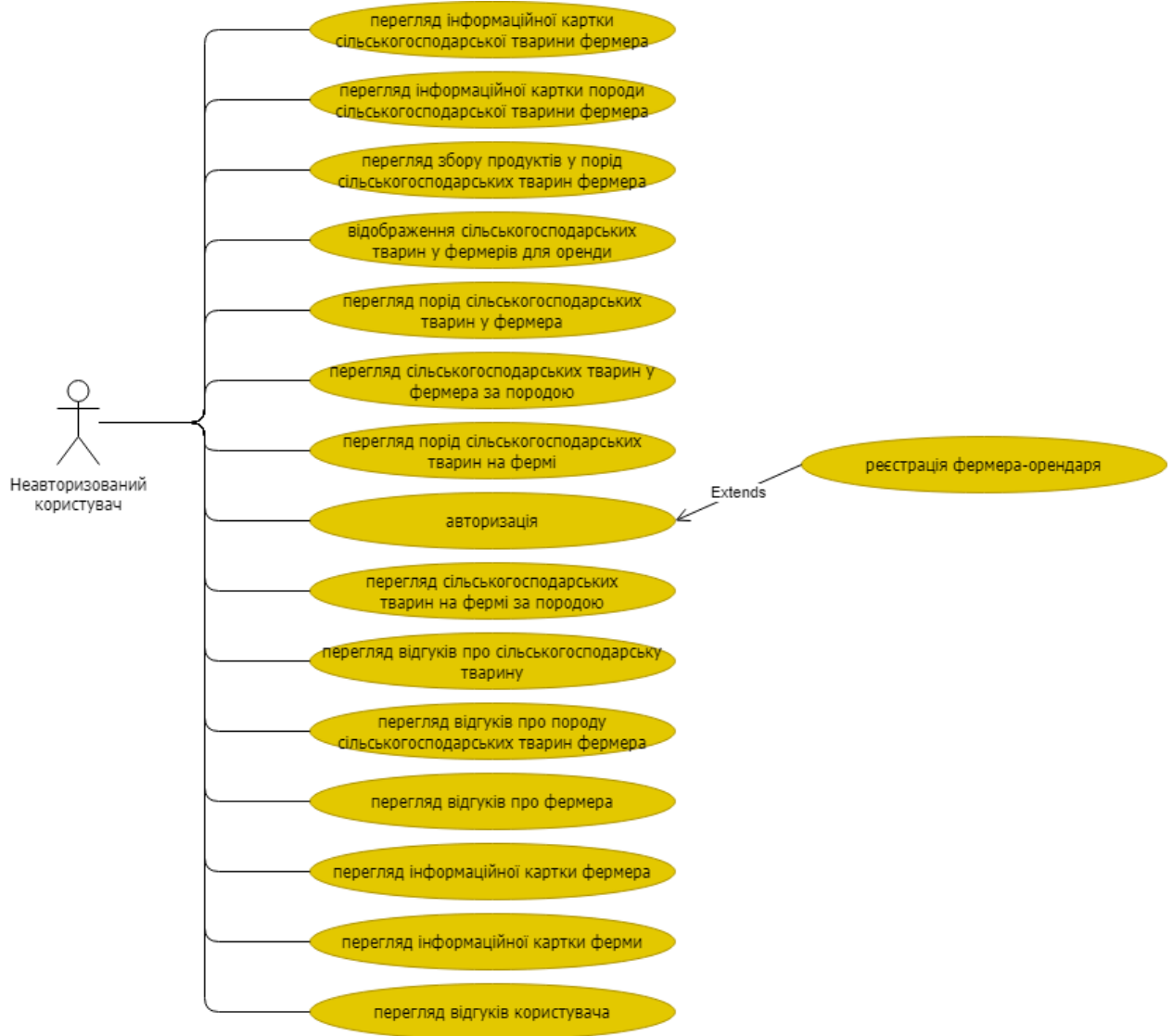


Рисунок 2.1 – Діаграма прецедентів програми оренди сільськогосподарських тварин для неавторизованого користувача

Для всього основного потоку неавторизованого користувача передбачено один альтернативний сценарій – реєстрація фермера-орендаря, що реалізується у випадку, коли фермер-орендар не може виконати авторизацію тому, що немає такого облікового запису фермера-орендаря, який би відповідав його потребам.

Фермер-власник у підсумку повинен працювати з таким основним потоком сценаріїв:

- внесення даних породи сільськогосподарської тварини фермера;
- внесення даних сільськогосподарської тварини фермера;

- керування замовленням власних сільськогосподарських тварин;
- перегляд поданих замовлень орендування сільськогосподарських тварин фермера;
- перегляд діючих орендувань сільськогосподарських тварин фермера;
- перегляд підтверджених замовлень орендування сільськогосподарських тварин фермера;
- перегляд скасованих замовлень орендування сільськогосподарських тварин фермера;
- перегляд завершених замовлень орендування сільськогосподарських тварин фермера;
- внесення збору продуктів у породи сільськогосподарської тварини фермером;
- визначення періоду оренди сільськогосподарської тварини;
- надання відгуку про фермера;
- редагування даних інформаційної картки ферми;
- редагування даних інформаційної картки фермера;
- збереження інформаційної картки фермера;
- збереження інформаційної картки породи сільськогосподарської тварини фермера;
- перегляд збережених інформаційних карток фермерів;
- перегляд збережених інформаційних карток порід сільськогосподарських тварин фермерів;
- перегляд замовлення орендування сільськогосподарських тварин у фермера;
- перегляд стану всіх власних тварин;
- перегляд історії зміни станів замовлення орендування сільськогосподарських тварин у фермера (рис. 2.2).

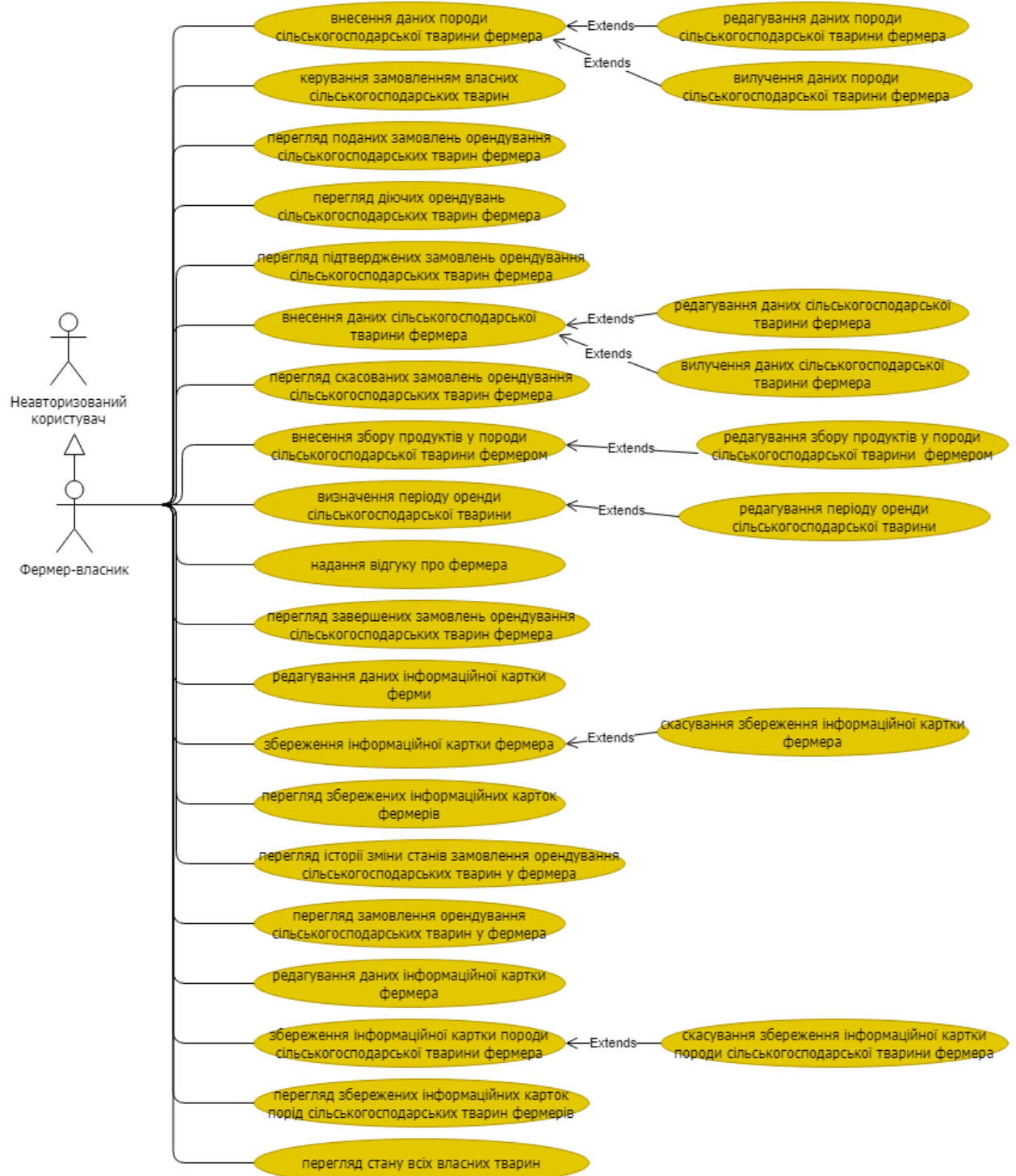


Рисунок 2.2 – Діаграма прецедентів програми оренди сільськогосподарських тварин для фермера-власника

Альтернативні сценарії фермера-власника:

– редагування даних породи сільськогосподарської тварини фермера, що є альтернативним сценарієм для внесення даних породи сільськогосподарської тварини фермера, коли дані породи

сільськогосподарської тварини фермера вже внесені, проте потрібно їх оновити;

- редагування даних сільськогосподарської тварини фермера, що є альтернативним сценарієм для внесення даних сільськогосподарської тварини фермера, коли дані сільськогосподарської тварини фермера вже внесені, проте потрібно їх оновити;

- вилучення даних сільськогосподарської тварини фермера, що є альтернативним сценарієм для внесення даних сільськогосподарської тварини фермера, коли дані сільськогосподарської тварини фермера вже внесені, проте така сільськогосподарська тварина фермера більше ніколи не буде доступною для оренди;

- вилучення даних породи сільськогосподарської тварини фермера, що є альтернативним сценарієм для внесення даних породи сільськогосподарської тварини фермера, коли дані породи сільськогосподарської тварини фермера вже внесені, проте така порода сільськогосподарської тварини фермера більше ніколи не буде доступною для оренди;

- скасування збереження інформаційної картки фермера, що є альтернативним сценарієм для збереження інформаційної картки фермера, коли збереження інформаційної картки фермера цим фермером-орендарем виконано, але більше не потрібне;

- скасування збереження інформаційної картки породи сільськогосподарської тварини фермера, що є альтернативним сценарієм для збереження інформаційної картки породи сільськогосподарської тварини фермера, коли збереження інформаційної картки породи сільськогосподарської тварини фермера цим фермером-орендарем виконано, але більше не потрібне;

- редагування збору продуктів у породи сільськогосподарської тварини фермером, що є альтернативним сценарієм для внесення збору продуктів у породи сільськогосподарської тварини фермером, коли дані збору продуктів у

породи сільськогосподарської тварини фермером вже внесені, проте потрібно їх оновити;

– редагування періоду оренди сільськогосподарської тварини, що є альтернативним сценарієм для визначення періоду оренди сільськогосподарської тварини, коли дані періоду оренди сільськогосподарської тварини вже внесені, проте потрібно їх оновити.

Фермер-орендар у підсумку повинен працювати з таким основним потоком сценаріїв:

- замовлення орендування сільськогосподарських тварин у фермера;
- перегляд власних поданих замовлень орендування сільськогосподарських тварин;
- перегляд власних діючих орендувань сільськогосподарських тварин;
- перегляд власних підтверджених замовлень орендування сільськогосподарських тварин;
- перегляд власних скасованих замовлень орендування сільськогосподарських тварин;
- перегляд власних завершених замовлень орендування сільськогосподарських тварин;
- перегляд збору продуктів у сільськогосподарських тварин в оренді;
- внесення збору продуктів у сільськогосподарської тварини в оренді;
- надання відгуку про сільськогосподарську тварину;
- надання відгуку про фермера;
- редагування даних інформаційної картки фермера;
- збереження інформаційної картки фермера;
- збереження інформаційної картки породи сільськогосподарської тварини фермера;
- перегляд збережених інформаційних карток фермерів;
- перегляд збережених інформаційних карток порід сільськогосподарських тварин фермерів;

- перегляд замовлення орендування сільськогосподарських тварин у фермера;
- перегляд історії зміни станів замовлення орендування сільськогосподарських тварин у фермера (рис. 2.3).

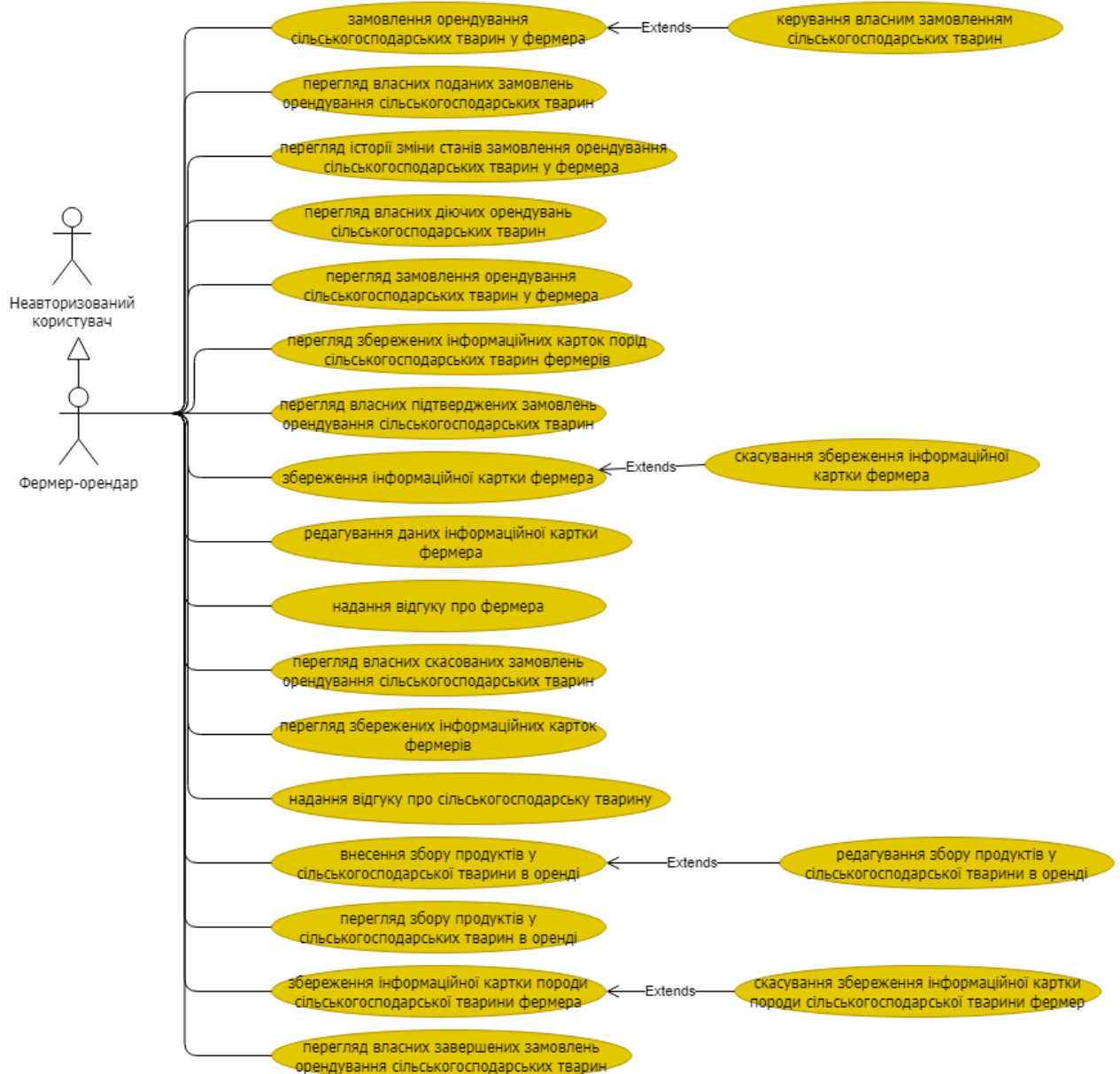


Рисунок 2.3 – Діаграма прецедентів програми оренди сільськогосподарських тварин для фермера-орендаря

Альтернативні сценарії фермера-орендаря:

- керування власним замовленням сільськогосподарських тварин, що є альтернативним сценарієм для замовлення орендування

сільськогосподарських тварин у фермера, коли це замовлення орендування сільськогосподарських тварин у фермера цим фермером-орендарем виконано, але він бажає вплинути на нього (переважно скасуванням);

- скасування збереження інформаційної картки породи сільськогосподарської тварини фермер, що є альтернативним сценарієм для збереження інформаційної картки породи сільськогосподарської тварини фермера, коли збереження інформаційної картки породи сільськогосподарської тварини фермера виконано, але більше потреби в цьому немає;

- скасування збереження інформаційної картки фермера, що є альтернативним сценарієм для збереження інформаційної картки фермера, коли збереження інформаційної картки фермера виконано, але більше потреби в цьому немає;

- редагування збору продуктів у сільськогосподарської тварини в оренді, що є альтернативним сценарієм для внесення збору продуктів у сільськогосподарської тварини в оренді, коли внесення збору продуктів у сільськогосподарської тварини в оренді виконано, проте потрібно його оновити.

2.3 Проектування структури бази даних

У структурі бази даних, створеної на основі системи MySQL (безкоштовна, вільна, надійна), передбачено ряд таблиць.

Таблиця `Farmanimaltype` необхідна для визначення видів сільськогосподарських тварин за стовпцями:

- `id` – ідентифікатор виду сільськогосподарських тварин;
- `farmanimaltypename` – назва виду сільськогосподарських тварин;
- `farmanimaltypephoto` – презентаційна фотографія виду сільськогосподарських тварин.

Таблиця Farmanimalsort необхідна для визначення порід сільськогосподарських тварин за стовпцями:

- id – ідентифікатор породи сільськогосподарських тварин;
- farmanimaltype – вид сільськогосподарських тварин, до якого належить ця порода, є зовнішнім ключем з таблиці Farmanimaltype, має тип один-до-багатьох;
- farmanimalsortname – назва породи сільськогосподарських тварин;
- farmanimalsortinfo – опис породи сільськогосподарських тварин;
- farmanimalsortphoto – презентаційна фотографія породи сільськогосподарських тварин.

Таблиця Farmanimal необхідна для визначення порід сільськогосподарських тварин у фермерів за стовпцями:

- id – ідентифікатор породи сільськогосподарської тварини у фермера;
- farm – ферма, на якій знаходиться сільськогосподарська тварина цієї породи у фермера, є зовнішнім ключем з таблиці Farm, має тип один-до-багатьох;
- farmanimalsort – порода сільськогосподарських тварин, якій належать ці тварини фермера, є зовнішнім ключем з таблиці Farmanimalsort, має тип один-до-багатьох;
- farmanimalquantity – кількість сільськогосподарських тварин цієї породи у цього фермера;
- farmanimalspecial – коментар щодо особливостей сільськогосподарських тварин цієї породи у цього фермера;
- farmanimalphoto – презентаційна фотографія сільськогосподарських тварин цієї породи у цього фермера;
- farmanimalscore – оцінка сільськогосподарських тварин цієї породи у цього фермера;
- farmrentalpriceweek – вартість доби оренди сільськогосподарських тварин цієї породи у цього фермера при оренді не менше ніж на тиждень;

– farmrentalpricemonth – вартість доби оренди сільськогосподарських тварин цієї породи у цього фермера при оренді не менше ніж на місяць;

– farmrentalpriceseason – вартість доби оренди сільськогосподарських тварин цієї породи у цього фермера при оренді не менше ніж на сезон (3 місяці);

– farmrentalpriceyear – вартість доби оренди сільськогосподарських тварин цієї породи у цього фермера при оренді не менше ніж на рік.

Таблиця Farmproduct необхідна для визначення продуктів, які можуть бути отримані від сільськогосподарських тварин (молоко, яйця тощо), за стовпцями:

– id – ідентифікатор продукту, який може бути отриманий від сільськогосподарських тварин;

– farmanimalproductname – назва продукту, який може бути отриманий від сільськогосподарських тварин.

Таблиця Farmanimalproduct необхідна для визначення статистики збору продуктів від сільськогосподарських тварин фермера за стовпцями:

– id – ідентифікатор статистичного запису збору продуктів від сільськогосподарських тварин фермера;

– farmproduct – продукт, який може бути отриманий від сільськогосподарських тварин, за яким зведено цю статистику щодо збору продуктів, є зовнішнім ключем з таблиці Farmproduct, має тип один-до-багатьох;

– farmanimal – порода сільськогосподарської тварини у фермера, за якою зведено цю статистику щодо збору продуктів, є зовнішнім ключем з таблиці Farmanimal, має тип один-до-багатьох;

– farmvolumeaverage – середній обсяг продуктів, який раніше було отримано від сільськогосподарських тварин фермера за цією породою;

– farmcalculated – визначення того, чи це значення було обчислено, чи воно було задано самим власником стосовно статистики збору продуктів від сільськогосподарських тварин фермера.

Таблиця `Farmanimalinstance` необхідна для визначення екземплярів сільськогосподарських тварин у фермера за стовпцями:

- `id` – ідентифікатор екземпляру сільськогосподарських тварин у фермера;
- `farmanimal` – порода сільськогосподарської тварини у фермера, до якої належить цей екземпляр тварини, є зовнішнім ключем з таблиці `Farmanimal`, має тип один-до-багатьох;
- `farmanimalname` – кличка екземпляру сільськогосподарських тварин у фермера (за наявності);
- `farmanimalscore` – середня оцінка цьому екземпляру сільськогосподарських тварин у фермера з відгуків орендарів;
- `farmanimalage` – вік сільськогосподарської тварини;
- `farmanimalstate` – стан, в якому перебуває ця сільськогосподарська тварина та оренда за нею;
- `farmanimalcolor` – колір цього екземпляру цієї сільськогосподарської тварини у фермера;
- `farmanimalspecial` – коментар щодо особливостей цього екземпляру цієї сільськогосподарської тварини у фермера.

Таблиця `Farmanimalinstancerentable` необхідна для визначення доступності екземпляру сільськогосподарської тварини фермера для оренди за стовпцями:

- `id` – ідентифікатор доступності екземпляру сільськогосподарської тварини фермера для оренди;
- `farmanimalinstance` – екземпляр сільськогосподарської тварини фермера, доступний для оренди, є зовнішнім ключем з таблиці `Farmanimalinstance`, має тип один-до-багатьох;
- `farmrentablefrom` – дата, з якої доступний цей екземпляр сільськогосподарської тварини фермера для оренди;
- `farmrentableto` – дата, до якої доступний цей екземпляр сільськогосподарської тварини фермера для оренди.

Таблиця `Farmanimalinstancephoto` необхідна для визначення фотографій екземплярів сільськогосподарських тварин у фермера за стовпцями:

- `id` – ідентифікатор фотографії екземпляра сільськогосподарських тварин у фермера;
- `farmanimalinstance` – екземпляр сільськогосподарської тварини у фермера, за якою зроблено фотографію, є зовнішнім ключем з таблиці `Farmanimalinstance`, має тип один-до-багатьох;
- `farmanimalinstancephotocontent` – вміст фотографії екземпляра сільськогосподарських тварин у фермера;
- `farmanimalinstancephototime` – дата, коли було зроблено фотографію екземпляра сільськогосподарських тварин у фермера.

Таблиця `Farmer` необхідна для визначення фермерів за стовпцями:

- `id` – ідентифікатор фермера;
- `farmerdjango` – акаунт користувача Django, за яким працює фермер у програмі, є зовнішнім ключем з таблиці `User`, має тип один-до-одного;
- `farmerphone` – номер телефону фермера;
- `farmerscore` – середня оцінка, виставлена орендарями сільськогосподарських тварин фермеру;
- `farmerphoto` – презентаційна фотографія фермера;
- `farmerinformation` – опис фермера.

Таблиця `Farm` необхідна для визначення ферм за стовпцями:

- `id` – ідентифікатор ферми;
- `farmer` – фермер, якому належить ця ферма, є зовнішнім ключем з таблиці `Farmer`, має тип один-до-багатьох;
- `farmregion` – регіон, де ця ферма знаходиться, є зовнішнім ключем з таблиці `Farmregion`, має тип один-до-багатьох;
- `farmaddress` – адреса ферми у цьому регіоні;
- `farmphoto` – презентаційна фотографія ферми;
- `farminformation` – опис ферми.

Таблиця Farmregion необхідна для визначення регіонів, де можуть знаходитись ферми, за стовпцями:

- id – ідентифікатор регіону, де може знаходитись ферма;
- farmregionname – назва регіону, де може знаходитись ферма;
- farmregioncountry – країна, де знаходиться регіон, де може знаходитись ферма.

Таблиця Farmrental необхідна для визначення замовлень оренд сільськогосподарських тварин за стовпцями:

- id – ідентифікатор замовлення оренди сільськогосподарських тварин у фермера;
- farmerowner – фермер, який володіє фермою, з якої замовлено оренду сільськогосподарських тварин, є зовнішнім ключем з таблиці Farmer, має тип один-до-багатьох;
- farmerrenter – фермер, який замовив оренду сільськогосподарських тварин, є зовнішнім ключем з таблиці Farmer, має тип один-до-багатьох;
- farmrentalprice – вартість замовлення оренди сільськогосподарських тварин у фермера;
- farmrentalstart – дата початку оренди сільськогосподарських тварин у фермера;
- farmrentalend – дата закінчення оренди сільськогосподарських тварин у фермера;
- farmrentalto – адреса, за якою має відбуватися оренда сільськогосподарських тварин у фермера, тобто куди вони мають бути перевезені.

Таблиця Farmrentalanimal необхідна для визначення складу замовлень сільськогосподарських тварин у фермера стосовно порід за стовпцями:

- id – ідентифікатор складу замовлення сільськогосподарських тварин у фермера;

– `farmrental` – замовлення сільськогосподарських тварин у фермера, за яким ця позиція складу встановлена, є зовнішнім ключем з таблиці `Farmrental`, має тип один-до-багатьох;

– `farmanimal` – порода сільськогосподарських тварин у фермера, яку замовлено для оренди, є зовнішнім ключем з таблиці `Farmanimal`, має тип один-до-багатьох;

– `farmanimalquantity` – кількість сільськогосподарських тварин цієї породи у фермера, яку було замовлено для оренди.

Таблиця `Farmrentalanimalinstance` необхідна для визначення складу замовлень сільськогосподарських тварин у фермера стосовно конкретних екземплярів за стовпцями:

– `id` – ідентифікатор складу замовлення сільськогосподарських тварин у фермера стосовно конкретного екземпляру;

– `farmrentalanimal` – склад замовлення сільськогосподарських тварин у фермера за породою, до якого належить цей екземпляр, є зовнішнім ключем з таблиці `Farmrentalanimal`, має тип один-до-багатьох;

– `farmanimalinstance` – конкретний екземпляр сільськогосподарських тварин у фермера, який було орендовано за цією породою, є зовнішнім ключем з таблиці `Farmanimalinstance`, має тип один-до-багатьох.

Таблиця `Farmrentalstatus` необхідна для визначення станів замовлення оренди сільськогосподарських тварин за стовпцями:

– `id` – ідентифікатор стану замовлення оренди сільськогосподарських тварин;

– `farmrental` – замовлення оренди сільськогосподарських тварин, за яким збережено цей стан, є зовнішнім ключем з таблиці `Farmrental`, має тип один-до-багатьох;

– `farmrentalstatusset` – стан замовлення оренди сільськогосподарських тварин, що збережено;

– `farmrentalstatustime` – час збереження цього стану замовлення оренди сільськогосподарських тварин;

– farmrentalstatusmemo – коментар до цього стану замовлення оренди сільськогосподарських тварин.

Таблиця Farmanimalinstancereview необхідна для визначення відгуків за екземплярами сільськогосподарських тварин у оренді за стовпцями:

– id – ідентифікатор відгуку за екземпляром сільськогосподарської тварини у оренді;

– farmanimalinstance – екземпляр сільськогосподарської тварини, за яким було створено відгук, є зовнішнім ключем з таблиці Farmanimalinstance, має тип один-до-багатьох;

– farmanimaldetalisation – визначення того, чи цей відгук за самим конкретним екземпляром сільськогосподарської тварини у оренді, чи за породою сільськогосподарських тварин у фермера;

– farmer – фермер, який цей відгук за екземпляром сільськогосподарської тварини створив, є зовнішнім ключем з таблиці Farmer, має тип один-до-багатьох;

– farmreviewtext – текст відгуку за екземпляром сільськогосподарської тварини;

– farmreviewscore – оцінка відгуку за екземпляром сільськогосподарської тварини;

– farmreviewtime – час відгуку за екземпляром сільськогосподарської тварини.

Таблиця Farmerreview необхідна для визначення відгуків про фермерів-власників за результатами оренди сільськогосподарських тварин за стовпцями:

– id – ідентифікатор відгуку про фермера-власника за результатами оренди сільськогосподарських тварин;

– farmerabout – фермер, про якого створено відгук за результатами оренди сільськогосподарських тварин, є зовнішнім ключем з таблиці Farmer, має тип один-до-багатьох;

– farmer – фермер, який створив відгук за результатами оренди сільськогосподарських тварин, є зовнішнім ключем з таблиці Farmer, має тип один-до-багатьох;

– farmreviewtext – текст відгуку про фермера-власника за результатами оренди сільськогосподарських тварин;

– farmreviewscore – оцінка відгуку про фермера-власника за результатами оренди сільськогосподарських тварин;

– farmreviewtime – час створення відгуку про фермера-власника за результатами оренди сільськогосподарських тварин.

Таблиця Farmanimalbook необхідна для визначення збереження інформаційної картки породи сільськогосподарської тварини фермера за стовпцями:

– id – ідентифікатор збереження інформаційної картки породи сільськогосподарської тварини фермера;

– farmanimal – порода сільськогосподарської тварини фермера, яку зберіг фермер, є зовнішнім ключем з таблиці Farmanimal, має тип один-до-багатьох;

– farmer – фермер, який зберіг породу сільськогосподарської тварини фермера, є зовнішнім ключем з таблиці Farmer, має тип один-до-багатьох;

– farmbookmemo – коментар до збереження інформаційної картки породи сільськогосподарської тварини фермера;

– farmbooktime – час збереження інформаційної картки породи сільськогосподарської тварини фермера.

Таблиця Farmerbook необхідна для визначення збереження інформаційної картки фермера за стовпцями:

– id – ідентифікатор збереження інформаційної картки фермера;

– farmeron – фермер, інформаційну картку якого було збережено, є зовнішнім ключем з таблиці Farmer, має тип один-до-багатьох;

– farmer – фермер, який зберіг інформаційну картку іншого фермера, є зовнішнім ключем з таблиці Farmer, має тип один-до-багатьох;

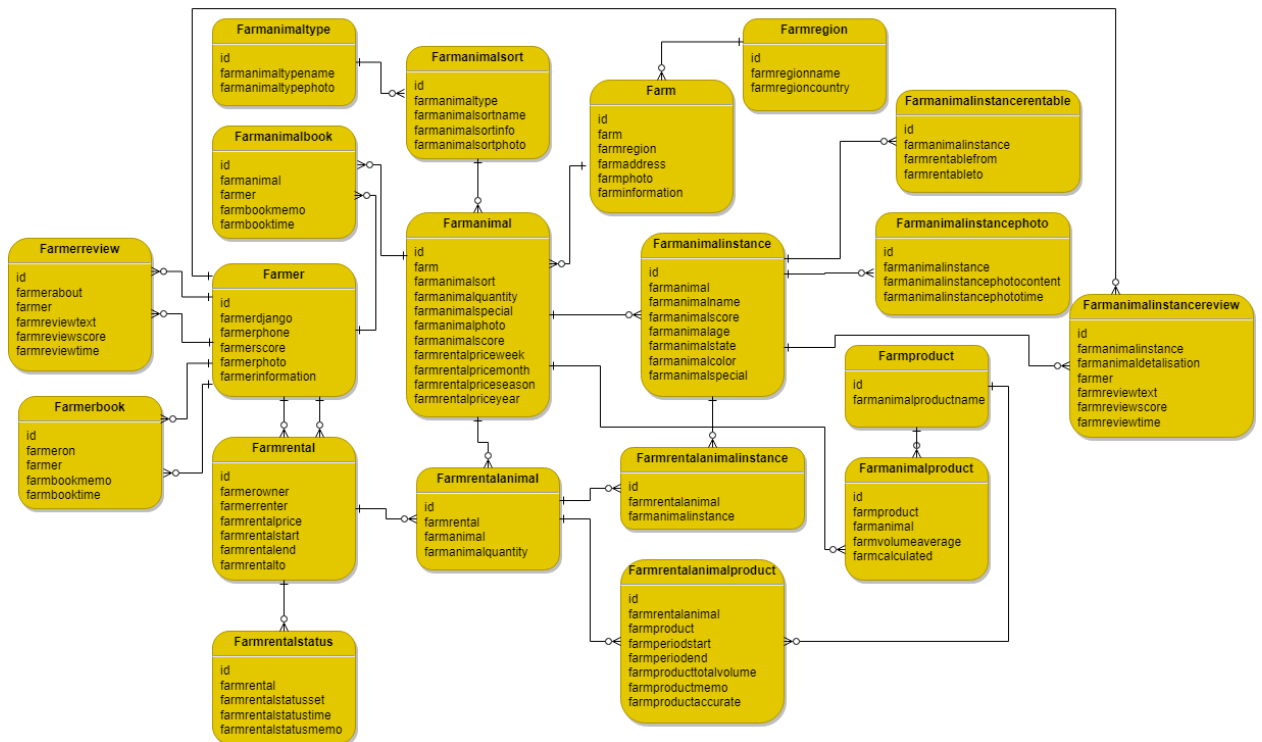


Рисунок 2.4 – Схема бази даних програми оренди сільськогосподарських тварин

Таблиця Farmrentalanimalproduct необхідна для визначення статистики збору орендарями продуктів з порід сільськогосподарських тварин у фермерів за стовпцями:

- `id` – ідентифікатор статистики збору орендарями продуктів з порід сільськогосподарських тварин у фермерів;
- `farmrentalanimal` – порода сільськогосподарської тварини у фермера з замовлення про оренду, за якою орендар визначив статистику збору продуктів, є зовнішнім ключем з таблиці `Farmrentalanimal`, має тип один-до-багатьох;
- `farmproduct` – продукт, який може бути отриманий від сільськогосподарських тварин, за яким зведено цю статистику щодо збору

продуктів, є зовнішнім ключем з таблиці Farmproduct, має тип один-до-багатьох;

- farmperiodstart – дата початку статистики збору орендарями продуктів з порід сільськогосподарських тварин у фермерів;

- farmperiodend – дата закінчення статистики збору орендарями продуктів з порід сільськогосподарських тварин у фермерів;

- farmproducttotalvolume – обсяг продуктів, які було зібрано орендарем з сільськогосподарських тварин цієї породи у фермера за вказаний період часу;

- farmproductmemo – коментар до особливостей збору орендарями продуктів з порід сільськогосподарських тварин у фермерів;

- farmproductaccurate – визначення того, чи це точно зібрані дані, чи вони можуть містити деякі неточності, наприклад, через пропущені дані.

2.4 Висновки за розділом 2

У розділі 2 пояснено зроблений вибір мови програмування Python, фреймворку Django, системи MySQL для розробки, визначено сценарії роботи фермера-власника, фермера-орендаря, неавторизованого користувача з програмою, сформовано структуру бази даних.

З РОЗРОБКА ПРОГРАМИ

3.1 Порядок роботи над функціями програми

Програма оренди сільськогосподарських тварин за звернення за адресою вебсайту без авторизації виконує відображення сільськогосподарських тварин у фермерів для оренди, надаючи далі можливість авторизуватись, а вже в залежності від ролі далі можна працювати з одним з розділів меню, обираючи потрібний (рис. 3.1).

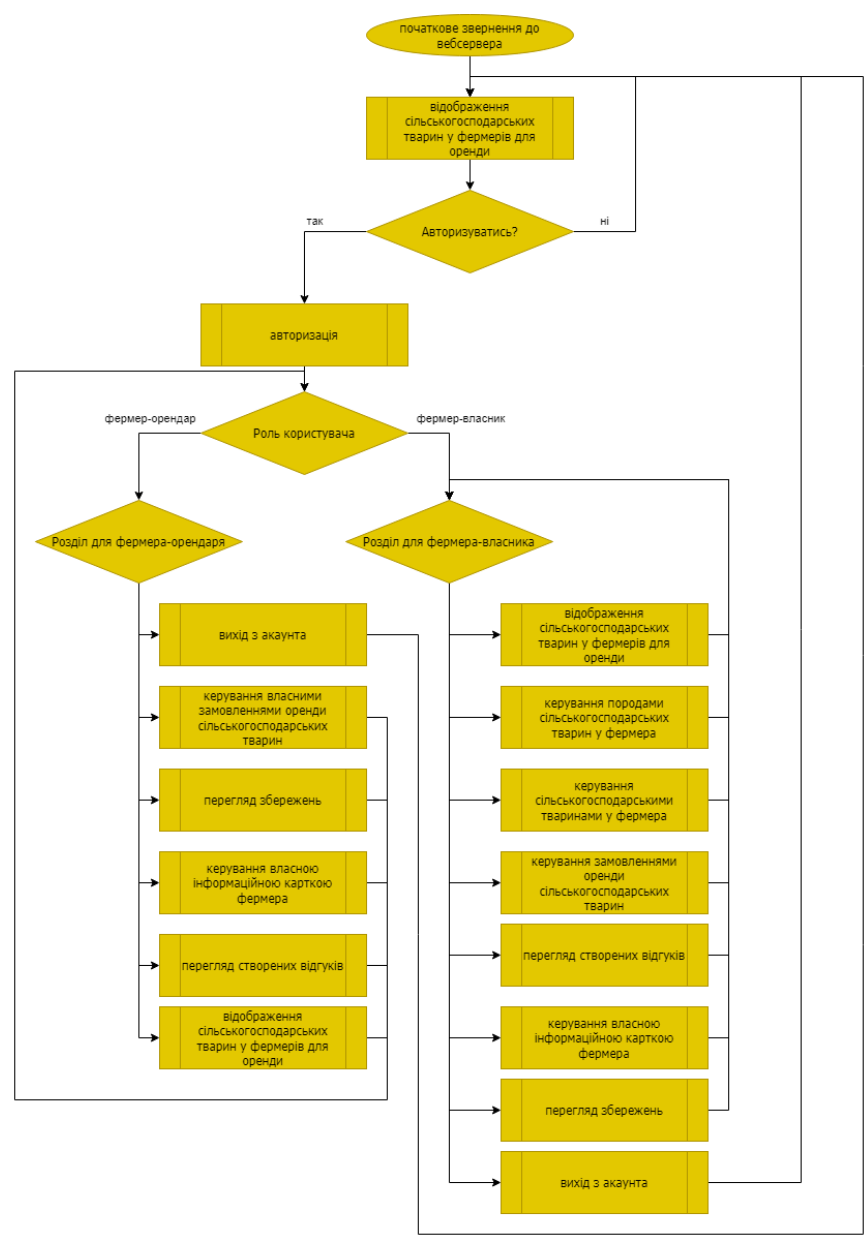


Рисунок 3.1 – Схема функціонування програми оренди сільськогосподарських тварин

3.2 Реалізація програми оренди сільськогосподарських тварин

Шаблон MVT, який лежить в основі побудови застосунків фреймворком Django, чітко визначив структуру застосунку, а відповідно і програмні одиниці, які в програмі оренди сільськогосподарських тварин були реалізовані. Нижче описані найбільш сумарно значущі одиниці створеної програми. Спочатку відображені розроблені класи.

Клас `Farmanimaltype` є моделлю визначення видів сільськогосподарських тварин.

Поле `farmanimaltypename` з класу `Farmanimaltype` є назвою виду сільськогосподарських тварин, реалізованою типом `CharField`.

Поле `farmanimaltypephoto` з класу `Farmanimaltype` є презентаційною фотографією виду сільськогосподарських тварин, реалізованою типом `ImageField`.

Клас `Farmanimalsort` є моделлю визначення порід сільськогосподарських тварин.

Поле `farmanimaltype` з класу `Farmanimalsort` є видом сільськогосподарських тварин, до якого належить ця порода, реалізованим типом `ForeignKey` з моделлю `Farmanimaltype`.

Поле `farmanimalsortname` з класу `Farmanimalsort` є назвою породи сільськогосподарських тварин, реалізованою типом `CharField`.

Поле `farmanimalsortinfo` з класу `Farmanimalsort` є описом породи сільськогосподарських тварин, реалізованим типом `CharField`.

Поле `farmanimalsortphoto` з класу `Farmanimalsort` є презентаційною фотографією породи сільськогосподарських тварин, реалізованою типом `ImageField`.

Клас `Farmanimal` є моделлю визначення порід сільськогосподарських тварин у фермерів.

Поле `farm` з класу `Farmanimal` є фермою, на якій знаходиться сільськогосподарська тварина цієї породи у фермера, реалізованою типом `ForeignKey` з моделлю `Farm`.

Поле `farmanimalsort` з класу `Farmanimal` є породою сільськогосподарських тварин, якій належать ці тварини фермера, реалізованою типом `ForeignKey` з моделлю `Farmanimalsort`.

Поле `farmanimalquantity` з класу `Farmanimal` є кількістю сільськогосподарських тварин цієї породи у цього фермера, реалізованою типом `PositiveSmallIntegerField`.

Поле `farmanimalspecial` з класу `Farmanimal` є коментарем щодо особливостей сільськогосподарських тварин цієї породи у цього фермера, реалізованим типом `CharField`.

Поле `farmanimalphoto` з класу `Farmanimal` є презентаційною фотографією сільськогосподарських тварин цієї породи у цього фермера, реалізованою типом `ImageField`.

Поле `farmanimalscore` з класу `Farmanimal` є оцінкою сільськогосподарських тварин цієї породи у цього фермера, реалізованою типом `FloatField`.

Поле `farmrentalpriceweek` з класу `Farmanimal` є вартістю доби оренди сільськогосподарських тварин цієї породи у цього фермера при оренді не менше ніж на тиждень, реалізованою типом `FloatField`.

Поле `farmrentalpricemonth` з класу `Farmanimal` є вартістю доби оренди сільськогосподарських тварин цієї породи у цього фермера при оренді не менше ніж на місяць, реалізованою типом `FloatField`.

Поле `farmrentalpriceseason` з класу `Farmanimal` є вартістю доби оренди сільськогосподарських тварин цієї породи у цього фермера при оренді не менше ніж на сезон (3 місяці), реалізованою типом `FloatField`.

Поле `farmrentalpriceyear` з класу `Farmanimal` є вартістю доби оренди сільськогосподарських тварин цієї породи у цього фермера при оренді не менше ніж на рік, реалізованою типом `FloatField`.

Клас `Farmproduct` є моделлю визначення продуктів, які можуть бути отримані від сільськогосподарських тварин.

Поле `farmanimalproductname` з класу `Farmproduct` є назвою продукту, який може бути отриманий від сільськогосподарських тварин, реалізованою типом `CharField`.

Клас `Farmanimalproduct` є моделлю визначення статистики збору продуктів від сільськогосподарських тварин фермера.

Поле `farmproduct` з класу `Farmanimalproduct` є продуктом, який може бути отриманий від сільськогосподарських тварин, за яким зведено цю статистику щодо збору продуктів, реалізованим типом `ForeignKey` з моделлю `Farmproduct`.

Поле `farmanimal` з класу `Farmanimalproduct` є породою сільськогосподарської тварини у фермера, за якою зведено цю статистику щодо збору продуктів, реалізованою типом `ForeignKey` з моделлю `Farmanimal`.

Поле `farmvolumeaverage` з класу `Farmanimalproduct` є середнім обсягом продуктів, який раніше було отримано від сільськогосподарських тварин фермера за цією породою, реалізованим типом `FloatField`.

Поле `farmcalculated` з класу `Farmanimalproduct` є визначенням того, чи це значення було обчислено, чи воно було задано самим власником стосовно статистики збору продуктів від сільськогосподарських тварин фермера, реалізованим типом `BooleanField`.

Клас `Farmanimalinstance` є моделлю визначення екземплярів сільськогосподарських тварин у фермера.

Поле `farmanimal` з класу `Farmanimalinstance` є породою сільськогосподарської тварини у фермера, до якої належить цей екземпляр тварини, реалізованою типом `ForeignKey` з моделлю `Farmanimal`.

Поле `farmanimalname` з класу `Farmanimalinstance` є кличкою екземпляру сільськогосподарських тварин у фермера, реалізованою типом `CharField`.

Поле `faranimalscore` з класу `Farmanimalinstance` є середньою оцінкою цьому екземпляру сільськогосподарських тварин у фермера з відгуків орендарів, реалізованою типом `FloatField`.

Поле `faranimalage` з класу `Farmanimalinstance` є віком сільськогосподарської тварини, реалізованою типом `PositiveSmallIntegerField`.

Поле `faranimalstate` з класу `Farmanimalinstance` є станом, в якому перебуває ця сільськогосподарська тварина та оренда за нею, реалізованою типом `CharField`.

Поле `faranimalcolor` з класу `Farmanimalinstance` є кольором цього екземпляру цієї сільськогосподарської тварини у фермера, реалізованим типом `CharField`.

Поле `faranimalspecial` з класу `Farmanimalinstance` є коментарем щодо особливостей цього екземпляру цієї сільськогосподарської тварини у фермера, реалізованим типом `CharField`.

Клас `Farmanimalinstancerentable` є моделлю визначення доступності екземпляру сільськогосподарської тварини фермера для оренди.

Поле `faranimalinstance` з класу `Farmanimalinstancerentable` є екземпляром сільськогосподарської тварини фермера, доступний для оренди, реалізованим типом `ForeignKey` з моделлю `Farmanimalinstance`.

Поле `farmrentablefrom` з класу `Farmanimalinstancerentable` є датою, з якої доступний цей екземпляр сільськогосподарської тварини фермера для оренди, реалізованою типом `DateField`.

Поле `farmrentableto` з класу `Farmanimalinstancerentable` є датою, до якої доступний цей екземпляр сільськогосподарської тварини фермера для оренди, реалізованою типом `DateField`.

Клас `Farmanimalinstancephoto` є моделлю визначення фотографій екземплярів сільськогосподарських тварин у фермера.

Поле `faranimalinstance` з класу `Farmanimalinstancephoto` є екземпляром сільськогосподарської тварини у фермера, за якою зроблено фотографію, реалізованим типом `ForeignKey` з моделлю `Farmanimalinstance`.

Поле `farmanimalinstancephotocontent` з класу `Farmanimalinstancephoto` є вмістом фотографії екземпляра сільськогосподарських тварин у фермера, реалізованим типом `ImageField`.

Поле `farmanimalinstancephototime` з класу `Farmanimalinstancephoto` є датою, коли було зроблено фотографію екземпляра сільськогосподарських тварин у фермера, реалізованою типом `DateField`.

Клас `Farmer` є моделлю визначення фермерів.

Поле `farmerdjango` з класу `Farmer` є акаунтом користувача Django, за яким працює фермер у програмі, реалізованим типом `OneToOneField` з моделлю `User`.

Поле `farmerphone` з класу `Farmer` є номером телефону фермера, реалізованим типом `CharField`.

Поле `farmerscore` з класу `Farmer` є середньою оцінкою, виставленою орендарями сільськогосподарських тварин фермеру, реалізованою типом `FloatField`.

Поле `farmerphoto` з класу `Farmer` є презентаційною фотографією фермера, реалізованою типом `ImageField`.

Поле `farmerinformation` з класу `Farmer` є описом фермера, реалізованим типом `TextField`.

Клас `Farm` є моделлю визначення ферм.

Поле `farmer` з класу `Farm` є фермером, якому належить ця ферма, є зовнішнім ключем з таблиці `Farmer`, реалізованим типом `ForeignKey` з моделлю `Farm`.

Поле `farmregion` з класу `Farm` є регіоном, де ця ферма знаходиться, реалізованим типом `ForeignKey` з моделлю `Farmregion`.

Поле `farmaddress` з класу `Farm` є адресою ферми у цьому регіоні, реалізованою типом `CharField`.

Поле `farmphoto` з класу `Farm` є презентаційною фотографією ферми, реалізованою типом `ImageField`.

Поле `farminformation` з класу `Farm` є описом ферми, реалізованим типом `TextField`.

Клас `Farmregion` є моделлю визначення регіонів, де можуть знаходитись ферми.

Поле `farmregionname` з класу `Farmregion` є назвою регіону, де може знаходитись ферма, реалізованою типом `CharField`.

Поле `farmregioncountry` з класу `Farmregion` є країною, де знаходиться регіон, де може знаходитись ферма, реалізованою типом `CharField`.

Клас `Farmrental` є моделлю визначення замовлень оренд сільськогосподарських тварин.

Поле `farmerowner` з класу `Farmrental` є фермером, який володіє фермою, з якої замовлено оренду сільськогосподарських тварин, реалізованим типом `ForeignKey` з моделлю `Farmer`.

Поле `farmerrenter` з класу `Farmrental` є фермером, який замовив оренду сільськогосподарських тварин, реалізованим типом `ForeignKey` з моделлю `Farmer`.

Поле `farmrentalprice` з класу `Farmrental` є вартістю замовлення оренди сільськогосподарських тварин у фермера, реалізованою типом `FloatField`.

Поле `farmrentalstart` з класу `Farmrental` є датою початку оренди сільськогосподарських тварин у фермера, реалізованою типом `DateField`.

Поле `farmrentalend` з класу `Farmrental` є датою закінчення оренди сільськогосподарських тварин у фермера, реалізованою типом `DateField`.

Поле `farmrentalto` з класу `Farmrental` є адресою, за якою має відбуватися оренда сільськогосподарських тварин у фермера, тобто куди вони мають бути перевезені, реалізованою типом `CharField`.

Клас `Farmrentalanimal` є моделлю визначення складу замовлень сільськогосподарських тварин у фермера стосовно порід.

Поле `farmrental` з класу `Farmrentalanimal` є замовленням сільськогосподарських тварин у фермера, реалізованим типом `ForeignKey` з моделлю `Farmrental`.

Поле `farmanimal` з класу `Farmrentalanimal` є породою сільськогосподарських тварин у фермера, реалізованою типом `ForeignKey` з моделлю `Farmanimal`.

Поле `farmanimalquantity` з класу `Farmrentalanimal` є кількістю сільськогосподарських тварин цієї породи у фермера, яку було замовлено для оренди, реалізованою типом `PositiveSmallIntegerField`.

Поле `farmanimalinstances` з класу `Farmrentalanimal` є складом замовлення сільськогосподарських тварин у фермера за породою, стосовно набору конкретних екземплярів, реалізованим типом `ManyToManyField` з моделлю `Farmanimalinstance` за типом багато-до-багатьох.

Клас `Farmrentalstatus` є моделлю визначення станів замовлення оренди сільськогосподарських тварин.

Поле `farmrental` з класу `Farmrentalstatus` є замовленням оренди сільськогосподарських тварин, за яким збережено цей стан, реалізованим типом `ForeignKey` з моделлю `Farmrental`.

Поле `farmrentalstatusset` з класу `Farmrentalstatus` є станом замовлення оренди сільськогосподарських тварин, що збережено, реалізованим типом `CharField`.

Поле `farmrentalstatustime` з класу `Farmrentalstatus` є часом збереження цього стану замовлення оренди сільськогосподарських тварин, реалізованим типом `DateTimeField`.

Поле `farmrentalstatusmemo` з класу `Farmrentalstatus` є коментарем до цього стану замовлення оренди сільськогосподарських тварин, реалізованим типом `CharField`.

Клас `Farmanimalinstancereview` є моделлю визначення відгуків за екземплярами сільськогосподарських тварин у оренді.

Поле `farmanimalinstance` з класу `Farmanimalinstancereview` є екземпляром сільськогосподарської тварини, за яким було створено відгук, реалізованим типом `ForeignKey` з моделлю `Farmanimalinstance`.

Поле `farmanimaldetailisation` з класу `Farmanimalinstancereview` є визначенням того, чи цей відгук за самим конкретним екземпляром сільськогосподарської тварини у оренді, чи за породою сільськогосподарських тварин у фермера, реалізованим типом `BooleanField`.

Поле `farmer` з класу `Farmanimalinstancereview` є фермером, який цей відгук за екземпляром сільськогосподарської тварини створив, реалізованим типом `ForeignKey` з моделлю `Farmer`.

Поле `farmreviewtext` з класу `Farmanimalinstancereview` є текстом відгуку за екземпляром сільськогосподарської тварини, реалізованим типом `CharField`.

Поле `farmreviewscore` з класу `Farmanimalinstancereview` є оцінкою відгуку за екземпляром сільськогосподарської тварини, реалізованим типом `PositiveSmallIntegerField`.

Поле `farmreviewtime` з класу `Farmanimalinstancereview` є часом відгуку за екземпляром сільськогосподарської тварини, реалізованим типом `DateTimeField`.

Клас `Farmerreview` є моделлю визначення відгуків про фермерів-власників за результатами оренди сільськогосподарських тварин.

Поле `farmerabout` з класу `Farmerreview` є фермером, про якого створено відгук за результатами оренди сільськогосподарських тварин, реалізованим типом `ForeignKey` з моделлю `Farmer`.

Поле `farmer` з класу `Farmerreview` є фермером, який створив відгук за результатами оренди сільськогосподарських тварин, реалізованим типом `ForeignKey` з моделлю `Farmer`.

Поле `farmreviewtext` з класу `Farmerreview` є текстом відгуку про фермера-власника за результатами оренди сільськогосподарських тварин, реалізованим типом `CharField`.

Поле `farmreviewscore` з класу `Farmerreview` є оцінкою відгуку про фермера-власника за результатами оренди сільськогосподарських тварин, реалізованою типом `PositiveSmallIntegerField`.

Поле `farmreviewtime` з класу `Farmerreview` є часом створення відгуку про фермера-власника за результатами оренди сільськогосподарських тварин, реалізованим типом `DateTimeField`.

Клас `Farmanimalbook` є моделлю визначення збереження інформаційної картки породи сільськогосподарської тварини фермера.

Поле `farmanimal` з класу `Farmanimalbook` є породою сільськогосподарської тварини фермера, яку зберіг фермер, реалізованою типом `ForeignKey` з моделлю `Farmanimal`.

Поле `farmer` з класу `Farmanimalbook` є фермером, який зберіг породу сільськогосподарської тварини фермера, реалізованим типом `ForeignKey` з моделлю `Farmer`.

Поле `farmbookmemo` з класу `Farmanimalbook` є коментарем до збереження інформаційної картки породи сільськогосподарської тварини фермера, реалізованим типом `CharField`.

Поле `farmbooktime` з класу `Farmanimalbook` є часом збереження інформаційної картки породи сільськогосподарської тварини фермера, реалізованим типом `CharField`.

Клас `Farmerbook` є моделлю визначення збереження інформаційної картки фермера.

Поле `farmeron` з класу `Farmerbook` є фермером, інформаційну картку якого було збережено, реалізованим типом `ForeignKey` з моделлю `Farmer`.

Поле `farmer` з класу `Farmerbook` є фермером, який зберіг інформаційну картку іншого фермера, реалізованим типом `ForeignKey` з моделлю `Farmer`.

Поле `farmbookmemo` з класу `Farmerbook` є коментарем до збереження інформаційної картки фермера, реалізованим типом `CharField`.

Поле `farmbooktime` з класу `Farmerbook` є часом збереження інформаційної картки фермера, реалізованим типом `DateTimeField`.

Клас `Farmrentalanimalproduct` є моделлю визначення статистики збору орендарями продуктів з порід сільськогосподарських тварин у фермерів.

Поле `farmrentalanimal` з класу `Farmrentalanimalproduct` є породою сільськогосподарської тварини у фермера з замовлення про оренду, за якою орендар визначив статистику збору продуктів, реалізованою типом `ForeignKey` з моделлю `Farmrentalanimal`.

Поле `farmproduct` з класу `Farmrentalanimalproduct` є продуктом, який може бути отриманий від сільськогосподарських тварин, за яким зведено цю статистику щодо збору продуктів, реалізованою типом `ForeignKey` з моделлю `Farmproduct`.

Поле `farmperiodstart` з класу `Farmrentalanimalproduct` є датою початку статистики збору орендарями продуктів з порід сільськогосподарських тварин у фермерів, реалізованою типом `DateField`.

Поле `farmperiodend` з класу `Farmrentalanimalproduct` є датою закінчення статистики збору орендарями продуктів з порід сільськогосподарських тварин у фермерів, реалізованою типом `DateField`.

Поле `farmproducttotalvolume` з класу `Farmrentalanimalproduct` є обсягом продуктів, які було зібрано орендарем з сільськогосподарських тварин цієї породи у фермера за вказаний період часу, реалізованим типом `DecimalField`.

Поле `farmproductmemo` з класу `Farmrentalanimalproduct` є коментарем до особливостей збору орендарями продуктів з порід сільськогосподарських тварин у фермерів, реалізованим типом `CharField`.

Поле `farmproductaccurate` з класу `Farmrentalanimalproduct` є визначенням того, чи це точно зібрані дані, чи вони можуть містити деякі неточності, реалізованим типом `BooleanField`.

Наступними описані створені функції представлень у відповідь на роботу з функціональними можливостями, які були заплановані. Кожне представлення має розроблений шаблон вебсторінки на основі вільного шаблону [8].

Перегляд інформаційної картки сільськогосподарської тварини фермера реалізовано функцією `farm_animal_instance_info_view` з номером

сільськогосподарської тварини фермера `farmanimalinstance_id`, заданим як параметр.

Перегляд інформаційної картки породи сільськогосподарської тварини фермера реалізовано функцією `farmanimal_info_view` з номером породи сільськогосподарської тварини фермера `farmanimal_id`, заданим як параметр.

Внесення даних породи сільськогосподарської тварини фермера реалізовано функцією `farmanimal_add_view` з класом `FarmanimalForm`, що забезпечує відповідну форму для внесення даних породи сільськогосподарської тварини.

Редагування даних породи сільськогосподарської тварини фермера реалізовано функцією `farmanimal_edit_view` з номером породи сільськогосподарської тварини фермера `farmanimal_id`, заданим як параметр.

Внесення даних сільськогосподарської тварини фермера реалізовано функцією `farmanimalinstance_add_view` з класом `FarmanimalinstanceForm`, що забезпечує відповідну форму для внесення даних сільськогосподарської тварини фермера.

Редагування даних сільськогосподарської тварини фермера реалізовано функцією `farmanimalinstance_edit_view` з номером сільськогосподарської тварини фермера `farmanimalinstance_id`, заданим як параметр.

Вилучення даних сільськогосподарської тварини фермера реалізовано функцією `farmanimalinstance_delete_view` з номером сільськогосподарської тварини фермера `farmanimalinstance_id`, заданим як параметр.

Вилучення даних породи сільськогосподарської тварини фермера реалізовано функцією `farmanimal_delete_view` з номером породи сільськогосподарської тварини фермера `farmanimal_id`, заданим як параметр.

Відображення сільськогосподарських тварин у фермерів для оренди реалізовано функцією `farmanimal_search_view`, яка має параметри: `farmanimaltype_id` – номер виду сільськогосподарських тварин, `farmanimalsort_id` – порода сільськогосподарських тварин, `farmregion_id` –

номер регіону, quantity – потрібна кількість таких тварин, farmanimalscore – мінімальна виставлена оцінка, maxfarmprice – максимальна допустима вартість оренди таких сільськогосподарських тварин, startrent – дата початку оренди, endrent – дата закінчення оренди.

Замовлення орендування сільськогосподарських тварин у фермера реалізовано функцією farmrental_add_view.

Керування замовленням сільськогосподарських тварин фермером реалізовано функцією farmrental_manage_view з номером замовлення farmrental_id, заданим як параметр.

Перегляд стану всіх власних тварин фермером реалізовано функцією farmer_animalinstances_view.

Перегляд збору продуктів у порід сільськогосподарських тварин фермера реалізовано функцією farmer_animalproducts_view з логіном фермера farmer_login, заданим як параметр. Подібно влаштовані інші представлення роботи зі зборами продуктів та графіками оренди.

Надання відгуку про сільськогосподарську тварину реалізовано функцією farmanimalinstance_review_add_view з номером сільськогосподарської тварини фермера farmanimalinstance_id.

Надання відгуку про фермера реалізовано функцією farmerreview_add_view з логіном фермера farmer_login, заданим як параметр.

Редагування даних інформаційної картки фермера реалізовано функцією farmer_edit_view.

Редагування даних інформаційної картки ферми реалізовано функцією farm_edit_view з номером ферми farm_id, заданим як параметр.

Перегляд порід сільськогосподарських тварин у фермера реалізовано функцією farmer_farmanimals_view з логіном фермера farmer_login, заданим як параметр.

Перегляд сільськогосподарських тварин у фермера за породою реалізовано функцією farm_animal_sort_instaces_view з номером породи

сільськогосподарських тварин у фермера `farmanimalsort_id`, заданим як параметр.

Збереження інформаційної картки фермера реалізовано функцією `farmer_book_view` з логіном фермера `farmer_login`, заданим як параметр. За наявності збереження цієї інформаційної картки фермера цим фермером реалізується скасування.

Збереження інформаційної картки породи сільськогосподарської тварини фермера реалізовано функцією `farmanimal_book_view` з номером породи сільськогосподарської тварини фермера `farmanimalsort_id`, заданим як параметр. За наявності збереження цієї інформаційної картки породи сільськогосподарської тварини фермера цим фермером реалізується скасування.

Перегляд збережених інформаційних карток фермерів реалізовано функцією `farmer_books_view`.

Перегляд збережених інформаційних карток порід сільськогосподарських тварин фермерів реалізовано функцією `farmanimal_books_view`.

Перегляд власних поданих замовлень орендування сільськогосподарських тварин реалізовано функцією `farmer_created_farmrentals_view`.

Перегляд власних діючих орендувань сільськогосподарських тварин реалізовано функцією `farmer_active_farmrentals_view`.

Перегляд власних підтверджених замовлень орендування сільськогосподарських тварин реалізовано функцією `farmer_approved_farmrentals_view`.

Перегляд власних скасованих замовлень орендування сільськогосподарських тварин реалізовано функцією `farmer_cancelled_farmrentals_view`.

Перегляд власних завершених замовлень орендування сільськогосподарських тварин реалізовано функцією `farmer_ended_farmrentals_view`.

Перегляд поданих замовлень орендування сільськогосподарських тварин фермера реалізовано функцією `created_farmrentals_view`.

Перегляд діючих орендувань сільськогосподарських тварин фермера реалізовано функцією `active_farmrentals_view`.

Перегляд підтверджених замовлень орендування сільськогосподарських тварин фермера реалізовано функцією `approved_farmrentals_view`.

Перегляд скасованих замовлень орендування сільськогосподарських тварин фермера реалізовано функцією `cancelled_farmrentals_view`.

Перегляд завершених замовлень орендування сільськогосподарських тварин фермера реалізовано функцією `ended_farmrentals_view`.

Перегляд порід сільськогосподарських тварин на фермі реалізовано функцією `farm_animal_sorts_view` з номером ферми `farm_id`, заданим як параметр.

Перегляд сільськогосподарських тварин на фермі за породою реалізовано функцією `farm_animal_sort_view` з номером ферми `farm_id` та номером породи сільськогосподарських тварин `farmanimalsort_id`, заданими як параметри.

Перегляд відгуків про сільськогосподарську тварину реалізовано функцією `farm_animal_instance_reviews_view` з номером сільськогосподарської тварини у фермера `farmanimalinstance_id`, заданим як параметр.

Перегляд відгуків про породу сільськогосподарських тварин фермера реалізовано функцією `farm_animal_sort_reviews_view` з номером породи сільськогосподарських тварин на фермі `farmanimalsort_id`, заданим як параметр.

Перегляд відгуків про фермера реалізовано функцією `on_farmer_reviews_view` з логіном фермера `farmer_login`, заданим як параметр.

Перегляд відгуків користувача реалізовано функцією `farmer_reviews_view` з логіном фермера-користувача `farmer_login`, заданим як параметр.

Перегляд інформаційної картки фермера реалізовано функцією `farmer_info_view` з логіном користувача фермера `farmer_login`, заданим як параметр.

Перегляд замовлення орендування сільськогосподарських тварин у фермера реалізовано функцією `farmrental_view` з номером замовлення `farmrental_id`, заданим як параметр.

Перегляд історії зміни станів замовлення орендування сільськогосподарських тварин у фермера реалізовано функцією `farmrental_statuses_view` з номером замовлення `farmrental_id`, заданим як параметр.

3.3 Висновки за розділом 3

Шаблон MVT, який лежить в основі побудови застосунків фреймворком Django, чітко визначив структуру застосунку, а відповідно і програмні одиниці, які в програмі оренди сільськогосподарських тварин були реалізовані як результат виконання цього розділу.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

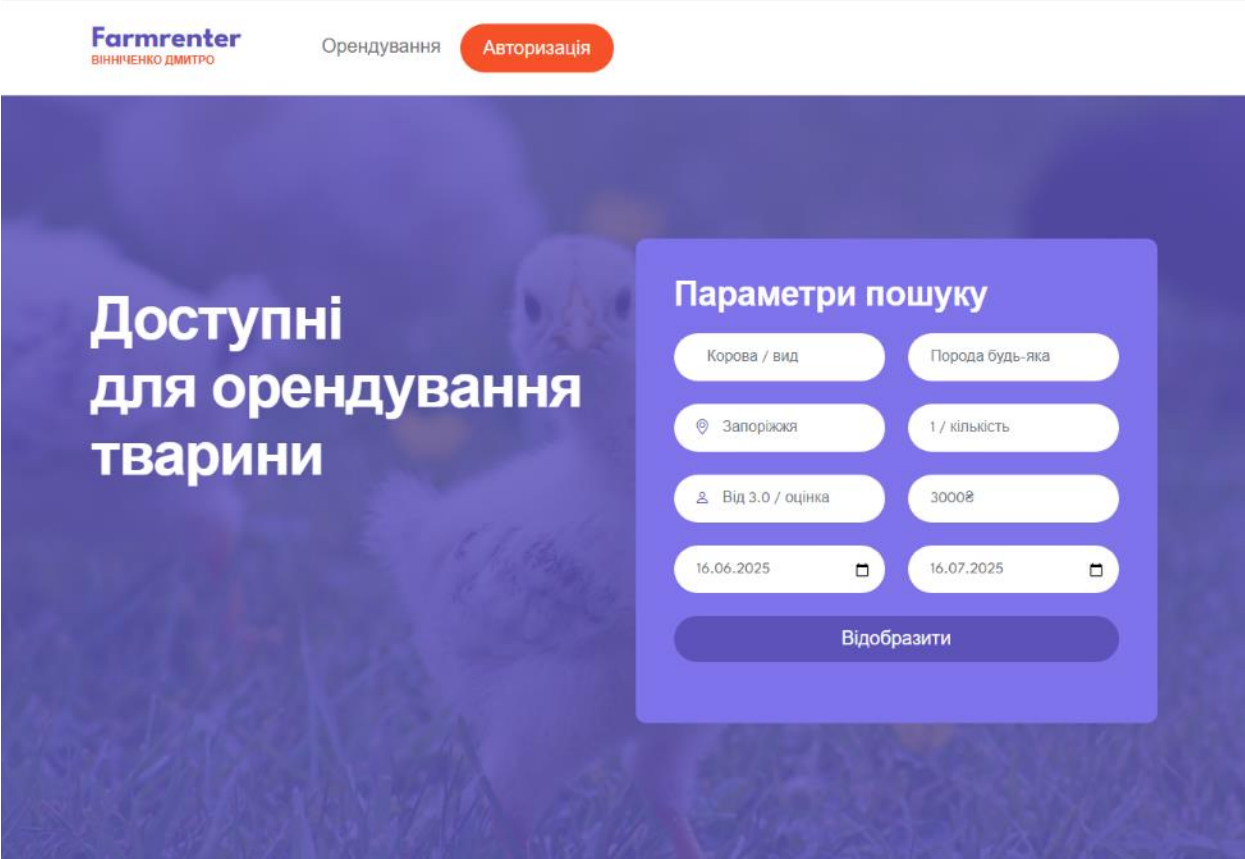
Програма призначена для оренди сільськогосподарських тварин у власників на певний період часу, надання власниками своїх сільськогосподарських тварин для оренди.

Програма надає можливості перегляду інформаційної картки сільськогосподарської тварини фермера, всієї породи, внесення, редагування і вилучення даних породи сільськогосподарської тварини фермера та конкретної тварини, відображення сільськогосподарських тварин у фермерів для оренди, замовлення орендування сільськогосподарських тварин у фермера, та керування цими замовленнями, перегляду стану всіх власних тварин фермером, перегляду збору продуктів у порід сільськогосподарських тварин фермера, перегляду власних замовлень орендування сільськогосподарських тварин та замовлень за власними сільськогосподарськими тваринами, перегляду, внесення, редагування збору продуктів у сільськогосподарських тварин в оренді та відповідних загальних даних фермером-власником, перегляду порід сільськогосподарських тварин у фермера, конкретних тварин за породою, порід сільськогосподарських тварин на фермі, конкретних сільськогосподарських тварин на фермі за породою, визначення і редагування періоду оренди сільськогосподарської тварини, надання відгуку про сільськогосподарську тварину, надання відгуку про фермера, перегляду відгуків про сільськогосподарську тварину, про породу сільськогосподарських тварин фермера, про фермера, перегляду відгуків користувача, перегляду і редагування даних інформаційної картки фермера, ферми, збереження інформаційної картки фермера, породи сільськогосподарської тварини фермера та скасування таких збережень, перегляду збережених інформаційних карток фермерів, порід сільськогосподарських тварин фермерів, перегляду замовлення орендування сільськогосподарських тварин у фермера, перегляду історії зміни станів

замовлення орендування сільськогосподарських тварин у фермера.

4.2 Опис роботи з програмою оренди сільськогосподарських тварин та тестування

Неавторизований користувач при першому зверненні до вебсервера отримує сторінку відображення сільськогосподарських тварин у фермерів для оренди, яка фактично є засобом пошуку (рис. 4.1). На цій сторінці неявна форма, яка дозволяє задати параметри пошуку, включаючи вид сільськогосподарської тварини, породу, регіон, де має знаходитись тварина, потрібну кількість тварин, мінімальну оцінку тварини, максимальну вартість оренди та строк оренди.



The screenshot displays the Farmrenter website interface. At the top, the logo 'Farmrenter' is accompanied by the text 'ВІННІЧЕНКО ДМИТРО'. Navigation links for 'Орендування' (Renting) and 'Авторизація' (Authorization) are present. The main content area features a large purple banner with the text 'Доступні для орендування тварини' (Animals available for rent). Overlaid on this banner is a search form titled 'Параметри пошуку' (Search parameters). The form includes several input fields: 'Корова / вид' (Cow / breed) with the value 'Порода будь-яка' (Any breed), a location field set to 'Запоріжжя' (Zaporizhzhia), a rating field set to 'Від 3.0 / оцінка' (From 3.0 / rating), a price field set to '3000€', and two date fields for the rental period: '16.06.2025' and '16.07.2025'. A 'Відобразити' (Display) button is located at the bottom of the search form.

Рисунок 4.1 – Параметри пошуку при відображенні сільськогосподарських тварин у фермерів для оренди

У результаті користувачу відображаються породи сільськогосподарських тварин у фермерів для оренди, які відповідають параметрам пошуку (рис. 4.2). За кожною породою відображається вартість такої оренди на вказані дати, регіон, порода, вид сільськогосподарської тварини. Присутнє посилання для переходу до сторінки інформаційної картки породи (посилання «Детально»), посилання для збереження породи до закладок, посилання для формування замовлення.

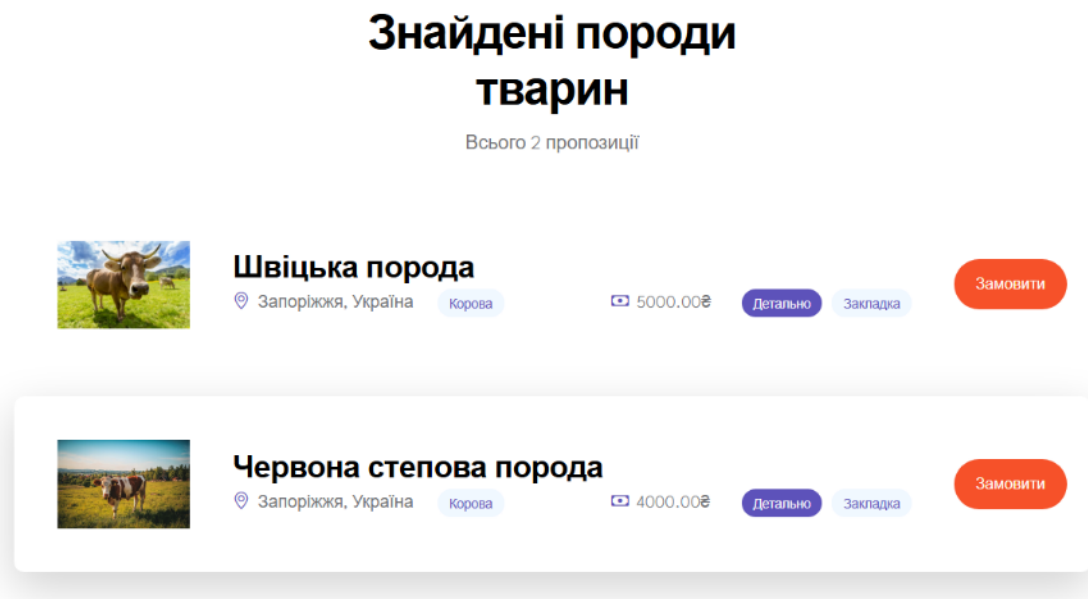


Рисунок 4.2 – Відображення сільськогосподарських тварин у фермерів для оренди

З інформаційної картки породи можна перейти до інформаційної картки ферми, а звідти до перегляду порід сільськогосподарських тварин на фермі (рис. 4.3). При цьому потрібно обрати вид сільськогосподарської тварини, натискаючи на одну з іконок (відображається білим кантуванням), а нижче відобразиться перелік порід. Також на сторінці надається інформація про фермера, про регіон, де знаходиться ферма.

Після авторизації фермер-власник має розділи меню, передбачені схемою функціонування програми оренди сільськогосподарських тварин.

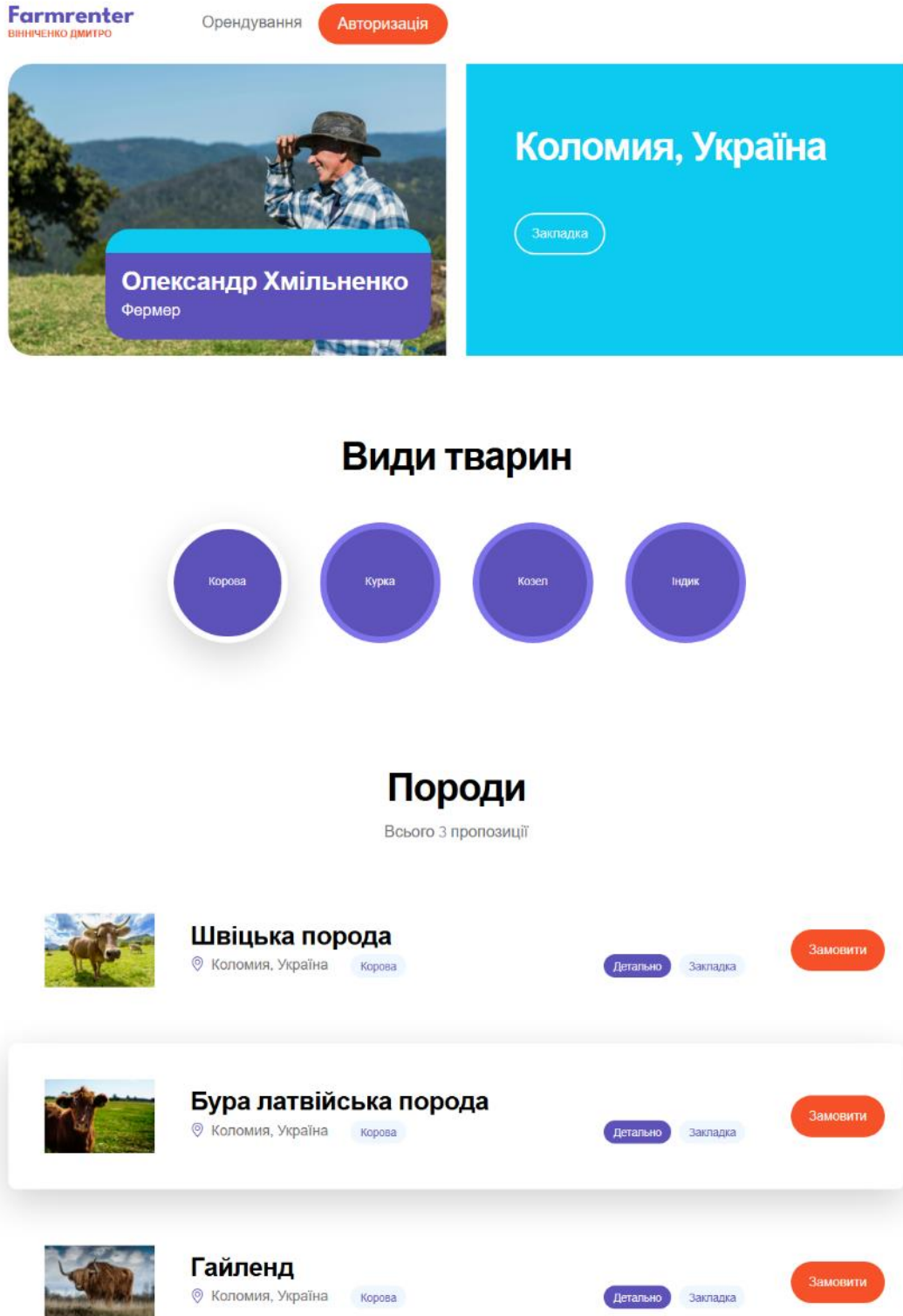


Рисунок 4.3 – Перегляд порід сільськогосподарських тварин на фермі

При виборі одного зі станів замовлень оренди сільськогосподарських тварин фермеру-власнику відображаються усі такі замовлення (рис. 4.4), а він може ними керувати або відкривати потрібне йому замовлення детально. З тієї

сторінки можна перейти до відображення історії зміни станів замовлення оренди сільськогосподарських тварин.

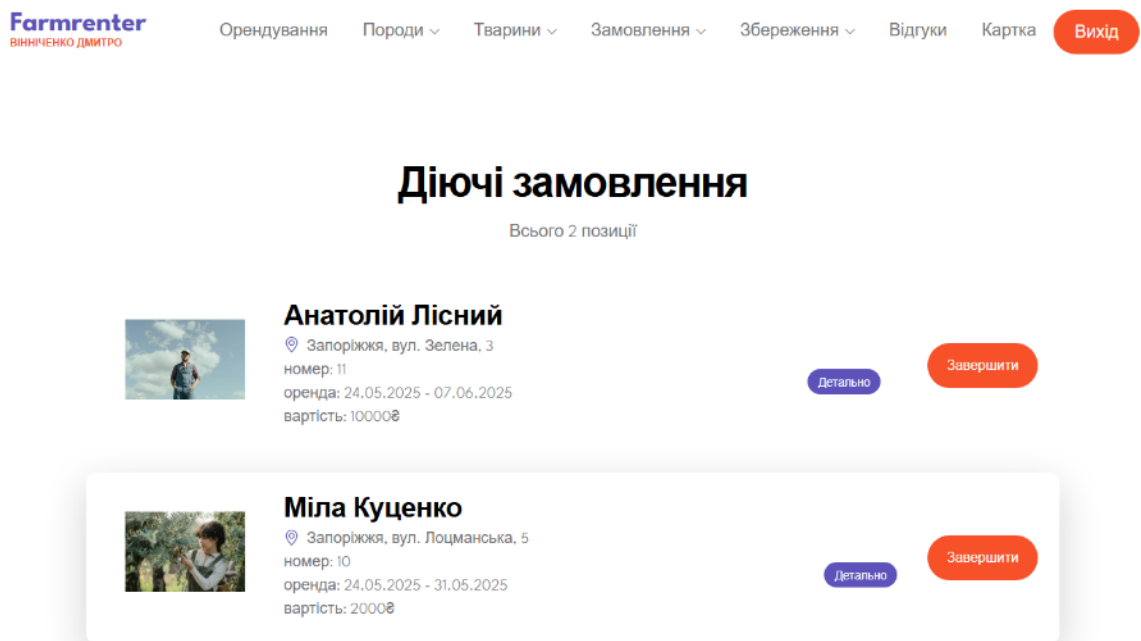


Рисунок 4.4 – Перегляд діючих орендувань сільськогосподарських тварин фермера

Фермеру-орендарю доступні подібні засоби, але без роботи з власним переліком порід тварин, тварин. Відповідно і при роботі з уже власними замовленнями доступні йому засоби обмежені в можливих станах (рис. 4.5).

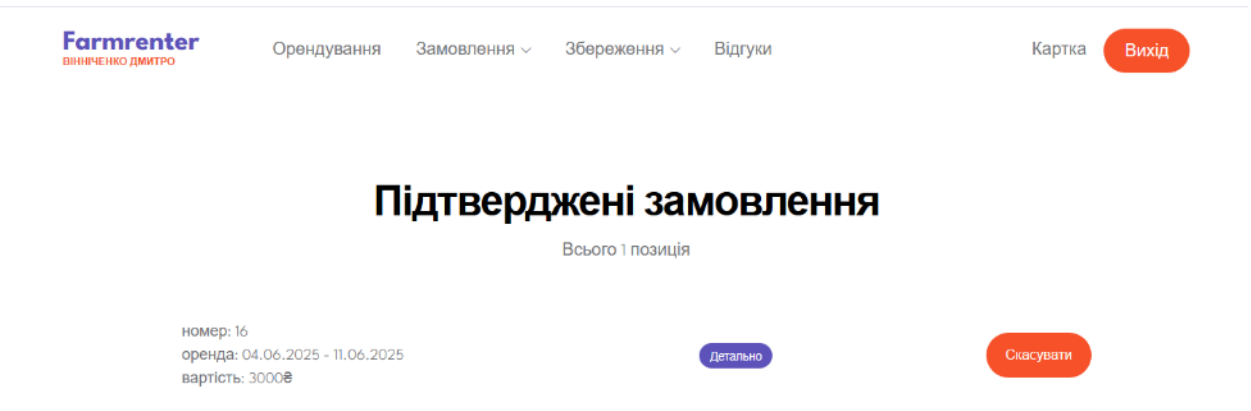


Рисунок 4.5 – Перегляд власних підтверджених замовлень орендування сільськогосподарських тварин

Усі приведені вище результати були отримані під час тестування програми орендування сільськогосподарських тварин, в якій було перевірено кожну функцію, що необхідно було реалізувати. Для кожної функції було отримано очікуваний результат. Усі результати є коректними. Тестування було успішним.

4.3 Висновки за розділом 4

У розділі проведено тестування програми оренди сільськогосподарських тварин з перевіркою результатів роботи кожної ролі користувача за кожною функціональною можливістю. Результати за всіма функціями виявились успішними.

ВИСНОВКИ

Програма оренди сільськогосподарських тварин створена на основі аналізу існуючого ринку, загальних тенденцій та тенденцій, які виникають під час коливань на ринку, а також аналізу існуючих програмних засобів для розв'язання цієї проблеми.

Реалізовані програмні одиниці застосунку Django для забезпечення всіх функцій програми за структурою шаблону MVT. Також для реалізації програми використано мову Python і систему керування базою даних MySQL.

Створена програма призначена для оренди сільськогосподарських тварин у власників на певний період часу, надання власниками своїх сільськогосподарських тварин для оренди. Програма надає можливості перегляду інформаційної картки сільськогосподарської тварини фермера, всієї породи, внесення, редагування і видалення даних породи сільськогосподарської тварини фермера та конкретної тварини, відображення сільськогосподарських тварин у фермерів для оренди, замовлення орендування сільськогосподарських тварин у фермера, та керування цими замовленнями, перегляду стану всіх власних тварин фермером, перегляду збору продуктів у порід сільськогосподарських тварин фермера, перегляду власних замовлень орендування сільськогосподарських тварин та замовлень за власними сільськогосподарськими тваринами, перегляду, внесення, редагування збору продуктів у сільськогосподарських тварин в оренді та відповідних загальних даних фермером-власником, перегляду порід сільськогосподарських тварин у фермера, конкретних тварин за породою, порід сільськогосподарських тварин на фермі, конкретних сільськогосподарських тварин на фермі за породою, визначення і редагування періоду оренди сільськогосподарської тварини, надання відгуку про сільськогосподарську тварину, надання відгуку про фермера, перегляду відгуків про сільськогосподарську тварину, про породу сільськогосподарських тварин фермера, про фермера, перегляду відгуків

користувача, перегляду і редагування даних інформаційної картки фермера, ферми, збереження інформаційної картки фермера, породи сільськогосподарської тварини фермера та скасування таких збережень, перегляду збережених інформаційних карток фермерів, порід сільськогосподарських тварин фермерів, перегляду замовлення орендування сільськогосподарських тварин у фермера, перегляду історії зміни станів замовлення орендування сільськогосподарських тварин у фермера.

Усе це в підсумку дозволило розробити ПЗ оренди сільськогосподарських тварин для надання можливості користувачам знаходити варіанти тимчасового володіння тваринами в господарських завданнях.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Rent The Chicken [Electronic resource]. – Access mode : <https://www.rentthechicken.com/>.
2. Trained Animals For Film. Rent Benay's Animals For Parties [Electronic resource]. – Access mode : <https://benaysanimals.com/>.
3. A-Z Animals [Electronic resource] : Hire Trained Animal Models for Advertising, Filming, Photography, PR, Events – A-Z Animals. – Access mode : <https://www.a-zanimals.co.uk/>.
4. Rent Mother Nature. Sustainable and Corporate Gifting [Electronic resource]. – Access mode : <https://rentmothernature.com/>.
5. Cow rental [Electronic resource] : Agriculture : Appenzellerland Tourismus. – Access mode : <https://www.appenzell.ch/en/landwirtschaft/landwirtschaft-erleben/cow-rental.html>.
6. Django Introduction [Electronic resource]. – Access mode : https://www.w3schools.com/django/django_intro.php.
7. CherryPy – A Minimalist Python Web Framework – CherryPy 0.1.dev54+gdca9548 documentation [Electronic resource]. – Access mode : <https://docs.cherrypy.dev/en/latest/>.
8. Gotto Job HTML Template by Tooplate [Electronic resource]. – Access mode : <https://www.tooplate.com/view/2134-gotto-job>.

ДОДАТОК А
Технічне завдання

Вступ

Володіння сільськогосподарськими тваринами є тривалим і відповідальним процесом, однак не всі люди готові стати учасниками цього процесу. З іншого боку оренда дозволяє спробувати будь-якому бажаючому те, на скільки він готовий зануритись у такий процес. Окрім того під час оренди можна збирати продукти, таким чином отримуючи певний економічний зиск. Наприклад, корови дозволяють отримати молоко, сир, курки – яйця. Усе це за певних умов, а особливо в складних економічних умовах, може бути економічно вигідно. Відповідно людина може на певний період часу стати учасником процесу володіння сільськогосподарськими тваринами, а потім це змінити, коли умови її життя змінились. При цьому тварина без проблем для її існування може повернутись до власника. Це змінює можливості людей, дозволяє взаємодіяти їм з природою у обмежений період часу. Застосування програмного забезпечення для реалізації таких задач є необхідною умовою в сучасному світі.

A.1 Підстави для розробки

Наказ № 209 від 28 квітня 2025 року був виданий у Національному університеті «Запорізька політехніка», визначивши для виконання цієї дипломної кваліфікаційної роботи бакалавра тему «Програмне забезпечення оренди сільськогосподарських тварин», яку було далі уточнено у вигляді завдання на дипломну кваліфікаційну роботу бакалавра.

A.2 Призначення розробки

Програма призначена для оренди сільськогосподарських тварин у власників на певний період часу, надання власниками своїх сільськогосподарських тварин для оренди.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програма оренди сільськогосподарських тварин повинна надавати функції:

- перегляд інформаційної картки сільськогосподарської тварини фермера;
- перегляд інформаційної картки породи сільськогосподарської тварини фермера;
- внесення даних породи сільськогосподарської тварини фермера;
- редагування даних породи сільськогосподарської тварини фермера;
- внесення даних сільськогосподарської тварини фермера;
- редагування даних сільськогосподарської тварини фермера;
- вилучення даних сільськогосподарської тварини фермера;
- вилучення даних породи сільськогосподарської тварини фермера;
- відображення сільськогосподарських тварин у фермерів для оренди;
- замовлення орендування сільськогосподарських тварин у фермера;
- керування замовленням сільськогосподарських тварин;
- перегляд стану всіх власних тварин фермером;
- перегляд збору продуктів у порід сільськогосподарських тварин фермера;
- перегляд власних поданих замовлень орендування сільськогосподарських тварин;
- перегляд власних діючих орендувань сільськогосподарських тварин;
- перегляд власних підтверджених замовлень орендування сільськогосподарських тварин;
- перегляд власних скасованих замовлень орендування сільськогосподарських тварин;
- перегляд власних завершених замовлень орендування сільськогосподарських тварин;

- перегляд поданих замовлень орендування сільськогосподарських тварин фермера;
- перегляд діючих орендувань сільськогосподарських тварин фермера;
- перегляд підтверджених замовлень орендування сільськогосподарських тварин фермера;
- перегляд скасованих замовлень орендування сільськогосподарських тварин фермера;
- перегляд завершених замовлень орендування сільськогосподарських тварин фермера;
- перегляд збору продуктів у сільськогосподарських тварин в оренді;
- внесення збору продуктів у сільськогосподарської тварини в оренді;
- редагування збору продуктів у сільськогосподарської тварини в оренді;
- внесення збору продуктів у породи сільськогосподарської тварини фермером-власником;
- редагування збору продуктів у породи сільськогосподарської тварини фермером-власником;
- перегляд порід сільськогосподарських тварин у фермера;
- перегляд сільськогосподарських тварин у фермера за породою;
- перегляд порід сільськогосподарських тварин на фермі;
- перегляд сільськогосподарських тварин на фермі за породою;
- визначення періоду оренди сільськогосподарської тварини;
- редагування періоду оренди сільськогосподарської тварини;
- надання відгуку про сільськогосподарську тварину;
- надання відгуку про фермера;
- перегляд відгуків про сільськогосподарську тварину;
- перегляд відгуків про породу сільськогосподарських тварин фермера;
- перегляд відгуків про фермера;
- перегляд відгуків користувача;
- реєстрація та авторизація;

- редагування даних інформаційної картки фермера;
- перегляд інформаційної картки фермера;
- редагування даних інформаційної картки ферми;
- перегляд інформаційної картки ферми;
- збереження інформаційної картки фермера;
- збереження інформаційної картки породи сільськогосподарської тварини фермера;
- перегляд збережених інформаційних карток фермерів;
- перегляд збережених інформаційних карток порід сільськогосподарських тварин фермерів;
- перегляд замовлення орендування сільськогосподарських тварин у фермера;
- перегляд історії зміни станів замовлення орендування сільськогосподарських тварин у фермера.

А.3.2 Умови експлуатації

Програма потребує для роботи наявності адміністратора з вміннями роботи з вебсервером, Django-застосунком через адміністративну панель та системою керування базами даних.

А.3.3 Вимоги до складу та параметрів технічних засобів

Мінімально сервер має бути забезпечений процесором на основі 2 ядер, 2 Гб оперативної пам'яті, 10 Гб вільного сховища жорсткого диску.

ДОДАТОК Б
Текст програми

```

from .models import Faranimalsort
from .models import Farmanimal
from .models import Farmanimaltype
from .models import Farmanimalproduct
from .models import Farmanimalinstance
from .models import Farmanimalinstancerentable
from .models import Farmproduct
from .models import Farmrentalanimalproduct
from .models import Farmanimalinstancephoto
from .models import Farmer
from .models import Farm
from .models import Farmregion
from .models import Farmrental
from .models import Farmrentalanimal
from .models import Farmrentalstatus
from .models import Farmanimalinstancereview
from .models import Farmerreview
from .models import Farmanimalbook
from .models import Farmerbook
from django.db.models import Q
from django.shortcuts import render

#перегляд власних поданих замовлень орендування
сільськогосподарських тварин
def farmer_created_farmrentals_view(request):
    if request.user.is_authenticated:
        try:
            farmer_auth = Farmer.objects.get(farmerdjango =
request.user)
        except Farmer.DoesNotExist:
            farmer_auth = None
        else:
            farmer_auth = None
        if not farmer_auth:
            return render(request, "farmrenter/error.html", {"mess":
"Перегляд власних поданих замовлень орендування можливий після
авторизації."})
        farmrental_statuses =
Farmrentalstatus.objects.select_related("farmrental",
"farmrental__farmerowner",
"farmrental__farmerrenter").filter(farmerrenter = farmer_auth,
farmrentalstatusset = "0")

```

```

        farmrental_statuses = farmrental_statuses.order_by("-
farmrentalstatustime")
        farmrentals = []
        for farmrental_status in farmrental_statuses:
            farm_rents = Farmrentalstatus.objects.filter(farmrental
= farmrental_status.farmrental, farmrentalstatuset__in = ["1", "2",
"3", "4"], farmrentalstatustime__gt =
farmrental_status.farmrentalstatustime)
            if not farm_rents:
                farmrentals.append(farmrental_status.farmrental)
        return render(request, "farmrenter/createdfarmrentals.html",
{"farmrentals": farmrentals})

#перегляд власних діючих орендувань сільськогосподарських тварин
def farmer_active_farmrentals_view(request):
    if request.user.is_authenticated:
        try:
            farmer_auth = Farmer.objects.get(farmerdjango =
request.user)
        except Farmer.DoesNotExist:
            farmer_auth = None
    else:
        farmer_auth = None
    if not farmer_auth:
        return render(request, "farmrenter/error.html", {"mess":
"Перегляд власних діючих замовлень орендування можливий після
авторизації."})
    farmrental_statuses =
Farmrentalstatus.objects.select_related("farmrental",
"farmrental__farmerowner",
"farmrental__farmerrenter").filter(farmerrenter = farmer_auth,
farmrentalstatuset = "3")
    farmrental_statuses = farmrental_statuses.order_by("-
farmrentalstatustime")
    farmrentals = []
    for farmrental_status in farmrental_statuses:
        farm_rents = Farmrentalstatus.objects.filter(farmrental
= farmrental_status.farmrental, farmrentalstatuset__in = ["2", "4"],
farmrentalstatustime__gt = farmrental_status.farmrentalstatustime)
        if not farm_rents:
            farmrentals.append(farmrental_status.farmrental)
    return render(request, "farmrenter/activefarmrentals.html",
{"farmrentals": farmrentals})

```

```

#перегляд власних підтверджених замовлень орендування
сільськогосподарських тварин
def farmer_approved_farmrentals_view(request):
    if request.user.is_authenticated:
        try:
            farmer_auth = Farmer.objects.get(farmerdjango =
request.user)
        except Farmer.DoesNotExist:
            farmer_auth = None
    else:
        farmer_auth = None
    if not farmer_auth:
        return render(request, "farmrenter/error.html", {"mess":
"Перегляд власних діючих замовлень орендування можливий після
авторизації."})
    farmrental_statuses =
Farmrentalstatus.objects.select_related("farmrental",
"farmrental__farmerowner",
"farmrental__farmerrenter").filter(farmerrenter = farmer_auth,
farmrentalstatusset = "1")
    farmrental_statuses = farmrental_statuses.order_by("-
farmrentalstatustime")
    farmrentals = []
    for farmrental_status in farmrental_statuses:
        farm_rents = Farmrentalstatus.objects.filter(farmrental
= farmrental_status.farmrental, farmrentalstatusset__in = ["2", "3",
"4"], farmrentalstatustime__gt =
farmrental_status.farmrentalstatustime)
        if not farm_rents:
            farmrentals.append(farmrental_status.farmrental)
    return render(request, "farmrenter/approvfarmrentals.html",
{"farmrentals": farmrentals})

#перегляд власних скасованих замовлень орендування
сільськогосподарських тварин
def farmer_cancelled_farmrentals_view(request):
    if request.user.is_authenticated:
        try:
            farmer_auth = Farmer.objects.get(farmerdjango =
request.user)
        except Farmer.DoesNotExist:
            farmer_auth = None
    else:

```

```

        farmer_auth = None
    if not farmer_auth:
        return render(request, "farmrenter/error.html", {"mess":
"Перегляд власних діючих замовлень орендування можливий після
авторизації."})
    farmrental_statuses =
Farmrentalstatus.objects.select_related("farmrental",
"farmrental__farmerowner",
"farmrental__farmerrenter").filter(farmerrenter = farmer_auth,
farmrentalstatusset = "2")
    farmrental_statuses = farmrental_statuses.order_by("-
farmrentalstatustime")
    farmrentals = []
    for farmrental_status in farmrental_statuses:
        farm_rents = Farmrentalstatus.objects.filter(farmrental
= farmrental_status.farmrental, farmrentalstatusset__in = ["1", "3",
"4"], farmrentalstatustime__gt =
farmrental_status.farmrentalstatustime)
        if not farm_rents:
            farmrentals.append(farmrental_status.farmrental)
        return render(request, "farmrenter/cancfarmrentals.html",
{"farmrentals": farmrentals})

#перегляд власних завершених замовлень орендування
сільськогосподарських тварин
def farmer_ended_farmrentals_view(request):
    if request.user.is_authenticated:
        try:
            farmer_auth = Farmer.objects.get(farmerdjango =
request.user)
        except Farmer.DoesNotExist:
            farmer_auth = None
    else:
        farmer_auth = None
    if not farmer_auth:
        return render(request, "farmrenter/error.html", {"mess":
"Перегляд власних діючих замовлень орендування можливий після
авторизації."})
    farmrental_statuses =
Farmrentalstatus.objects.select_related("farmrental",
"farmrental__farmerowner",
"farmrental__farmerrenter").filter(farmerrenter = farmer_auth,
farmrentalstatusset = "4")

```

```

        farmrental_statuses = farmrental_statuses.order_by("-
farmrentalstatustime")
        farmrentals = []
        for farmrental_status in farmrental_statuses:
            farm_rents = Farmrentalstatus.objects.filter(farmrental
= farmrental_status.farmrental, farmrentalstatusset__in = ["1", "2",
"3", "4"], farmrentalstatustime__gt =
farmrental_status.farmrentalstatustime)
            if not farm_rents:
                farmrentals.append(farmrental_status.farmrental)
        return render(request, "farmrenter/endfarmrentals.html",
{"farmrentals": farmrentals})

#перегляд поданих замовлень орендування сільськогосподарських
тварин фермера
def created_farmrentals_view(request):
    if request.user.is_authenticated:
        try:
            farmer_auth = Farmer.objects.get(farmerdjango =
request.user)
        except Farmer.DoesNotExist:
            farmer_auth = None
    else:
        farmer_auth = None
    if not farmer_auth:
        return render(request, "farmrenter/error.html", {"mess":
"Перегляд поданих замовлень орендування можливий після авторизації."})
    farmrental_statuses =
Farmrentalstatus.objects.select_related("farmrental",
"farmrental__farmerowner",
"farmrental__farmerrenter").filter(farmerowner = farmer_auth,
farmrentalstatusset = "0")
    farmrental_statuses = farmrental_statuses.order_by("-
farmrentalstatustime")
    farmrentals = []
    for farmrental_status in farmrental_statuses:
        farm_rents = Farmrentalstatus.objects.filter(farmrental
= farmrental_status.farmrental, farmrentalstatusset__in = ["1", "2",
"3", "4"], farmrentalstatustime__gt =
farmrental_status.farmrentalstatustime)
        if not farm_rents:
            farmrentals.append(farmrental_status.farmrental)

```

```

        return render(request,
            "farmrenter/farmcreatedfarmrentals.html", {"farmrentals":
farmrentals})

#перегляд діючих орендувань сільськогосподарських тварин фермера
def active_farmrentals_view(request):
    if request.user.is_authenticated:
        try:
            farmer_auth = Farmer.objects.get(farmerdjango =
request.user)
        except Farmer.DoesNotExist:
            farmer_auth = None
    else:
        farmer_auth = None
    if not farmer_auth:
        return render(request, "farmrenter/error.html", {"mess":
"Перегляд діючих орендувань можливий після авторизації."})
    farmrental_statuses =
Farmrentalstatus.objects.select_related("farmrental",
"farmrental__farmerowner",
"farmrental__farmerrenter").filter(farmerowner = farmer_auth,
farmrentalstatusset = "3")
    farmrental_statuses = farmrental_statuses.order_by("-
farmrentalstatustime")
    farmrentals = []
    for farmrental_status in farmrental_statuses:
        farm_rents = Farmrentalstatus.objects.filter(farmrental
= farmrental_status.farmrental, farmrentalstatusset__in = ["2", "4"],
farmrentalstatustime__gt = farmrental_status.farmrentalstatustime)
        if not farm_rents:
            farmrentals.append(farmrental_status.farmrental)
    return render(request,
        "farmrenter/farmactivefarmrentals.html", {"farmrentals": farmrentals})

#перегляд підтверджених замовлень орендування
сільськогосподарських тварин фермера
def approved_farmrentals_view(request):
    if request.user.is_authenticated:
        try:
            farmer_auth = Farmer.objects.get(farmerdjango =
request.user)
        except Farmer.DoesNotExist:
            farmer_auth = None

```

```

else:
    farmer_auth = None
    if not farmer_auth:
        return render(request, "farmrenter/error.html", {"mess":
"Перегляд підтверджених замовлень орендування можливий після
авторизації."})
    farmrental_statuses =
Farmrentalstatus.objects.select_related("farmrental",
"farmrental__farmerowner",
"farmrental__farmerrenter").filter(farmerowner = farmer_auth,
farmrentalstatusset = "1")
    farmrental_statuses = farmrental_statuses.order_by("-
farmrentalstatustime")
    farmrentals = []
    for farmrental_status in farmrental_statuses:
        farm_rents = Farmrentalstatus.objects.filter(farmrental
= farmrental_status.farmrental, farmrentalstatusset__in = ["2", "3",
"4"], farmrentalstatustime__gt =
farmrental_status.farmrentalstatustime)
        if not farm_rents:
            farmrentals.append(farmrental_status.farmrental)
    return render(request,
"farmrenter/farmapprovfarmrentals.html", {"farmrentals": farmrentals})

#перегляд скасованих замовлень орендування сільськогосподарських
тварин фермера
def cancelled_farmrentals_view(request):
    if request.user.is_authenticated:
        try:
            farmer_auth = Farmer.objects.get(farmerdjango =
request.user)
        except Farmer.DoesNotExist:
            farmer_auth = None
    else:
        farmer_auth = None
    if not farmer_auth:
        return render(request, "farmrenter/error.html", {"mess":
"Перегляд скасованих замовлень орендування можливий після
авторизації."})
    farmrental_statuses =
Farmrentalstatus.objects.select_related("farmrental",
"farmrental__farmerowner",

```

```

"farmrental__farmerrenter").filter(farmerowner = farmer_auth,
farmrentalstatusset = "2")
    farmrental_statuses = farmrental_statuses.order_by("-
farmrentalstatustime")
    farmrentals = []
    for farmrental_status in farmrental_statuses:
        farm_rents = Farmrentalstatus.objects.filter(farmrental
= farmrental_status.farmrental, farmrentalstatusset__in = ["1", "3",
"4"], farmrentalstatustime__gt =
farmrental_status.farmrentalstatustime)
        if not farm_rents:
            farmrentals.append(farmrental_status.farmrental)
    return render(request,
"farmrenter/farmcancfarmrentals.html", {"farmrentals": farmrentals})

#перегляд завершених замовлень орендування сільськогосподарських
тварин фермера
def ended_farmrentals_view(request):
    if request.user.is_authenticated:
        try:
            farmer_auth = Farmer.objects.get(farmerdjango =
request.user)
        except Farmer.DoesNotExist:
            farmer_auth = None
    else:
        farmer_auth = None
    if not farmer_auth:
        return render(request, "farmrenter/error.html", {"mess":
"Перегляд завершених замовлень орендування можливий після
авторизації."})
    farmrental_statuses =
Farmrentalstatus.objects.select_related("farmrental",
"farmrental__farmerowner",
"farmrental__farmerrenter").filter(farmerowner = farmer_auth,
farmrentalstatusset = "4")
    farmrental_statuses = farmrental_statuses.order_by("-
farmrentalstatustime")
    farmrentals = []
    for farmrental_status in farmrental_statuses:
        farm_rents = Farmrentalstatus.objects.filter(farmrental
= farmrental_status.farmrental, farmrentalstatusset__in = ["1", "2",
"3", "4"], farmrentalstatustime__gt =
farmrental_status.farmrentalstatustime)

```

```

        if not farm_rents:
            farmrentals.append(farmrental_status.farmrental)
        return render(request, "farmrenter/farmendfarmrentals.html",
{"farmrentals": farmrentals})

#перегляд історії зміни станів замовлення орендування
сільськогосподарських тварин у фермера
def farmrental_statuses_view(request, farmrental_id):
    if request.user.is_authenticated:
        try:
            farmer_auth = Farmer.objects.get(farmerdjango =
request.user)
        except Farmer.DoesNotExist:
            farmer_auth = None
    else:
        farmer_auth = None
    if not farmer_auth:
        return render(request, "farmrenter/error.html", {"mess":
"Перегляд історії зміни станів замовлення орендування
сільськогосподарських тварин у фермера можливий після авторизації."})
    farmrental =
Farmrental.objects.select_related("farmrental__farmerrenter").get(id =
farmrental_id, farmerowner = farmer_auth)
    if farmrental:
        farmrental_statuses =
Farmrentalstatus.objects.select_related("farmrental").filter(farmrenta
l__id = farmrental_id, farmrental__farmerowner = farmer_auth)
        farmrental_statuses = farmrental_statuses.order_by("-
farmrentalstatustime")
        return render(request,
"farmrenter/farmrentalstatuses.html", {"farmrental": farmrental,
"statuses": farmrental_statuses, "type": "farmer" })
    else:
        farmrental =
Farmrental.objects.select_related("farmrental__farmerrenter").get(id =
farmrental_id, farmerrenter = farmer_auth)
        if farmrental:
            farmrental_statuses =
Farmrentalstatus.objects.select_related("farmrental").filter(farmrenta
l__id = farmrental_id, farmrental__farmerrenter = farmer_auth)
            farmrental_statuses =
farmrental_statuses.order_by("-farmrentalstatustime")

```

```

        return render(request,
            "farmrenter/farmrentalstatuses.html", {"farmrental": farmrental,
            "statuses": farmrental_statuses, "type": "owner" })
    else:
        return render(request,
            "farmrenter/farmrentalstatuses.html", {"farmrental": None, "statuses":
            [], "type": "owner" })

#перегляд замовлення орендування сільськогосподарських тварин у
фермера
def farmrental_view(request, farmrental_id):
    if request.user.is_authenticated:
        try:
            farmer_auth = Farmer.objects.get(farmerdjango =
request.user)
        except Farmer.DoesNotExist:
            farmer_auth = None
        else:
            farmer_auth = None
        if not farmer_auth:
            return render(request, "farmrenter/error.html", {"mess":
"Перегляд замовлення орендування можливий після авторизації."})
        farmrental =
Farmrental.objects.select_related("farmrental__farmerrenter").get(id =
farmrental_id, farmerowner = farmer_auth)
        if farmrental:
            farmrental_status_last =
Farmrentalstatus.objects.select_related("farmrental").filter(farmrenta
l__id = farmrental_id, farmrental__farmerowner = farmer_auth)
            farmrental_status_last =
farmrental_status_last.order_by("-farmrentalstatustime")
            farmrental_status_last = farmrental_status_last[0]
            farmrental_animals =
Farmrentalanimal.objects.prefetch_related("farmanimalinstances").filte
r(farmrental = farmrental)
            for farmrental_animal in farmrental_animals:
                farmrental_animal.inbook = None
                if farmer_auth:
                    farmanimal_in_book =
Farmanimalbook.objects.filter(farmer = farmer_auth, farmanimal =
farmrental_animal.farmanimal)
                    if farmanimal_in_book:

```

```

        farmrental_animal.inbook =
farmanimal_in_book
    else:
        farmanimalinstances =
farmrental_animals.farmanimalinstances
        for farmanimalinstance in farmanimalinstances:
            farmanimalinstance.photos =
Farmanimalinstancephoto.objects.filter(farmanimalinstance =
farmanimalinstance)
            farmrental_animal.instances = farmanimalinstances
        return render(request, "farmrenter/farmrental.html",
{"farmrental": farmrental, "status": farmrental_status_last, "type":
"farmer", "animals": farmrental_animals })
    else:
        farmrental =
Farmrental.objects.select_related("farmrental__farmerrenter").get(id =
farmrental_id, farmerrenter = farmer_auth)
        if farmrental:
            farmrental_status_last =
Farmrentalstatus.objects.select_related("farmrental").filter(farmrenta
l__id = farmrental_id, farmrental__farmerrenter = farmer_auth)
            farmrental_status_last =
farmrental_status_last.order_by("-farmrentalstatustime")
            farmrental_status_last = farmrental_status_last[0]
            farmrental_animals =
Farmrentalanimal.objects.prefetch_related("farmanimalinstances").fille
r(farmrental = farmrental)
            for farmrental_animal in farmrental_animals:
                farmrental_animal.inbook = None
                if farmer_auth:
                    farmanimal_in_book =
Farmanimalbook.objects.filter(farmer = farmer_auth, farmanimal =
farmrental_animal.farmanimal)
                    if farmanimal_in_book:
                        farmrental_animal.inbook =
farmanimal_in_book
                farmanimalinstances =
farmrental_animals.farmanimalinstances
                for farmanimalinstance in farmanimalinstances:
                    farmanimalinstance.photos =
Farmanimalinstancephoto.objects.filter(farmanimalinstance =
farmanimalinstance)

```

```

        farmrental_animal.instances =
farmanimalinstances
        return render(request, "farmrenter/farmrental.html",
{"farmrental": farmrental, "status": farmrental_status_last, "type":
"owner", "animals": farmrental_animals })
    else:
        return render(request, "farmrenter/farmrental.html",
{"farmrental": None, "status": None, "type": "owner", "animals": [] })

#перегляд порід сільськогосподарських тварин на фермі
def farm_animal_sorts_view(request, farm_id):
    if request.user.is_authenticated:
        try:
            farmer_auth = Farmer.objects.get(farmerdjango =
request.user)
        except Farmer.DoesNotExist:
            farmer_auth = None
        else:
            farmer_auth = None
        try:
            farm_found = Farm.objects.select_related("farmer",
"farmregion").get(id = farm_id)
        except Farm.DoesNotExist:
            farm_found = None
        if not farm_found:
            return render(request, "farmrenter/error.html", {"mess":
"Про таку ферму нам нічого не відомо."})
        farmanimals =
Farmanimal.objects.select_related("farmanimalsort",
"farmanimalsort__farmanimaltype").filter(farm__id =
farm_id).values("farmanimalsort").distinct()
        for farmanimal in farmanimals:
            farmanimal.book = None
            if farmer_auth:
                farmanimal_in_book =
Farmanimalbook.objects.filter(farmanimal = farmanimal, farmer =
farmer_auth)
                if farmanimal_in_book:
                    farmanimal.book = farmanimal_in_book
            farmer_book = None
            if farmer_auth:
                farmer_in_book = Farmerbook.objects.filter(farmeron =
farm_found.farmer, farmer = farmer_auth)

```

```

        if farmer_in_book:
            farmer_book = farmer_in_book
        farmanimaletypes = Farmanimaltype.objects.all()
        return render(request, "farmrenter/farmsorts.html",
{"sorts": farmanimals, "farm": farm_found, "book": farmer_book,
"animals": farmanimaletypes })

#перегляд сільськогосподарських тварин на фермі за породою
def farm_animal_sort_view(request, farm_id, farmanimalsort_id):
    if request.user.is_authenticated:
        try:
            farmer_auth = Farmer.objects.get(farmerdjango =
request.user)
        except Farmer.DoesNotExist:
            farmer_auth = None
        else:
            farmer_auth = None
        try:
            farm_found = Farm.objects.select_related("farmer",
"farmregion").get(id = farm_id)
        except Farm.DoesNotExist:
            farm_found = None
        if not farm_found:
            return render(request, "farmrenter/error.html", {"mess":
"Про таку ферму нам нічого не відомо."})
        try:
            farm_animal_sort_found =
Farmanimalsort.objects.select_related("farmanimaltype").get(id =
farmanimalsort_id)
        except Farmanimalsort.DoesNotExist:
            farm_animal_sort_found = None
        if not farm_animal_sort_found:
            return render(request, "farmrenter/error.html", {"mess":
"Про таку породу сільськогосподарських тварин нам нічого не відомо."})
        farmanimals =
Farmanimal.objects.select_related("farmanimalsort",
"farmanimalsort__farmanimaltype").filter(farm__id = farm_id,
farmanimalsort__id = farmanimalsort_id)
        for farmanimal in farmanimals:
            farmanimal.instances =
Farmanimalinstance.objects.filter(farmanimal = farmanimal)
            farmanimal.book = None
            if farmer_auth:

```

```

        farmanimal_in_book =
Farmanimalbook.objects.filter(farmanimal = farmanimal, farmer =
farmer_auth)
        if farmanimal_in_book:
            farmanimal.book = farmanimal_in_book
        farmer_book = None
        if farmer_auth:
            farmer_in_book = Farmerbook.objects.filter(farmeron =
farm_found.farmer, farmer = farmer_auth)
            if farmer_in_book:
                farmer_book = farmer_in_book
        return render(request, "farmrenter/farmsort.html", {"sorts":
farmanimals, "farm": farm_found, "book": farmer_book, "animals":
farmanimals, "sort": farm_animal_sort_found })

#перегляд відгуків про сільськогосподарську тварину
def farm_animal_instance_reviews_view(request,
farmanimalinstance_id):
    if request.user.is_authenticated:
        try:
            farmer_auth = Farmer.objects.get(farmerdjango =
request.user)
        except Farmer.DoesNotExist:
            farmer_auth = None
    else:
        farmer_auth = None
    try:
        farm_animal_instance_found =
Farmanimalinstance.objects.select_related("farmanimal",
"farmanimal__farmanimalsort",
"farmanimal__farmanimalsort__farmanimaltype", "farmanimal__farm",
"farmanimal__farm__farmregion", "farmanimal__farm__farmer").get(id =
farmanimalinstance_id)
    except Farmanimalinstance.DoesNotExist:
        farm_animal_instance_found = None
    if not farm_animal_instance_found:
        return render(request, "farmrenter/error.html", {"mess":
"Про таку тварину нам нічого не відомо."})
    farmanimal_reviews =
Farmanimalinstancereview.objects.select_related("farmer",
"farmer__farmerdjango").filter(farmanimalinstance__id =
farmanimalinstance_id)

```

```

        farmanimal_reviews = farmanimal_reviews.order_by("-
farmreviewtime")
        for farmanimal_review in farmanimal_reviews:
            farmer_in_book = None
            if farmer_auth:
                farmer_in_book = Farmerbook.objects.filter(farmeron
= farmanimal_review.farmer, farmer = farmer_auth)
                if farmer_in_book:
                    farmanimal_review.book = farmer_in_book
                else:
                    farmanimal_review.book = None
            else:
                farmanimal_review.book = None
        farmer_book = None
        if farmer_auth:
            farmer_in_book = Farmerbook.objects.filter(farmeron =
farm_animal_instance_found.farmanimal.farm.farmer, farmer =
farmer_auth)
            if farmer_in_book:
                farmer_book = farmer_in_book
        farm_animal_book = None
        if farmer_auth:
            farm_animal_in_book =
Farmanimalbook.objects.filter(farmanimal =
farm_animal_instance_found.farmanimal, farmer = farmer_auth)
            if farm_animal_in_book:
                farm_animal_book = farm_animal_in_book
        return render(request, "farmrenter/instreviews.html",
{"reviews": farmanimal_reviews, "farmerbook": farmer_book,
"animalbook": farm_animal_book, "animal":
farm_animal_instance_found})

```

```

def farm_animal_sort_reviews_view(request, farmanimalsort_id):
    if request.user.is_authenticated:
        try:
            farmer_auth = Farmer.objects.get(farmerdjango =
request.user)
        except Farmer.DoesNotExist:
            farmer_auth = None
    else:
        farmer_auth = None
    try:

```

```

        farm_animal_sort_found =
Farmanimal.objects.select_related("farmanimalsort",
"farmanimalsort__farmanimaltype").get(id = farmanimalsort_id)
    except Farmanimal.DoesNotExist:
        farm_animal_sort_found = None
    if not farm_animal_sort_found:
        return render(request, "farmrenter/error.html", {"mess":
"Про таку породу сільськогосподарських тварин фермера нам нічого не
Відомо."})
    farmanimal_reviews =
Farmanimalinstancereview.objects.select_related("farmer",
"farmer__farmerdjango").filter(farmanimalinstance__farmanimal =
farm_animal_sort_found)
    farmanimal_reviews = farmanimal_reviews.order_by("-
farmreviewtime")
    for farmanimal_review in farmanimal_reviews:
        farmer_in_book = None
        if farmer_auth:
            farmer_in_book = Farmerbook.objects.filter(farmeron
= farmanimal_review.farmer, farmer = farmer_auth)
            if farmer_in_book:
                farmanimal_review.book = farmer_in_book
            else:
                farmanimal_review.book = None
        else:
            farmanimal_review.book = None
        farmer_book = None
        if farmer_auth:
            farmer_in_book = Farmerbook.objects.filter(farmeron =
farm_animal_instance_found.farmanimal.farm.farmer, farmer =
farmer_auth)
            if farmer_in_book:
                farmer_book = farmer_in_book
        farm_animal_book = None
        if farmer_auth:
            farm_animal_in_book =
Farmanimalbook.objects.filter(farmanimalinstance__farmanimal =
farm_animal_sort_found, farmer = farmer_auth)
            if farm_animal_in_book:
                farm_animal_book = farm_animal_in_book
        return render(request, "farmrenter/ansortreviews.html",
{"reviews": farmanimal_reviews, "farmerbook": farmer_book,
"animalbook": farm_animal_book, "animal": farm_animal_sort_found })

```

```

def on_farmer_reviews_view(request, farmer_login):
    if request.user.is_authenticated:
        try:
            farmer_auth = Farmer.objects.get(farmerdjango =
request.user)
        except Farmer.DoesNotExist:
            farmer_auth = None
    else:
        farmer_auth = None
    try:
        farmer_to =
Farmer.objects.select_related("farmerdjango").get(farmerdjango__userna
me = farmer_login)
    except Farmer.DoesNotExist:
        farmer_to = None
    if not farmer_to:
        return render(request, "farmrenter/error.html", {"mess":
"Про такого фермера нам нічого не відомо."})
    farmer_reviews =
Farmerreview.objects.select_related("farmer",
"farmer__farmerdjango").filter(farmerabout = farmer_to)
    farmer_reviews = farmer_reviews.order_by("-farmreviewtime")
    for farmer_review in farmer_reviews:
        farmer_review.book = None
        if farmer_auth:
            farmer_in_book = Farmerbook.objects.filter(farmeron
= farmer_review.farmer, farmer = farmer_auth)
            if farmer_in_book:
                farmer_review.book = farmer_in_book
        farmer_book = None
        if farmer_auth:
            farmer_in_book = Farmerbook.objects.filter(farmeron =
farmer_to, farmer = farmer_auth)
            if farmer_in_book:
                farmer_book = farmer_in_book
    return render(request, "farmrenter/farmerreviews.html",
{"reviews": farmer_reviews, "farmerbook": farmer_book, "farmer":
farmer_to })

def farmer_reviews_view(request, farmer_login = None):
    if request.user.is_authenticated:
        try:

```

```

        farmer_auth = Farmer.objects.get(farmerdjango =
request.user)
    except Farmer.DoesNotExist:
        farmer_auth = None
    else:
        farmer_auth = None
    if not farmer_login:
        farmer_from = farmer_auth
    else:
        try:
            farmer_from =
Farmer.objects.select_related("farmerdjango").get(farmerdjango__userna
me = farmer_login)
        except Farmer.DoesNotExist:
            farmer_from = None
        if not farmer_from:
            return render(request, "farmrenter/error.html",
{"mess": "Про такого фермера нам нічого не відомо."})
        farmer_reviews =
Farmerreview.objects.select_related("farmerabout",
"farmerabout__farmerdjango").filter(farmer = farmer_from)
        farmer_reviews = farmer_reviews.order_by("-farmreviewtime")
        for farmer_review in farmer_reviews:
            farmer_review.book = None
            if farmer_auth:
                farmer_in_book = Farmerbook.objects.filter(farmeron
= farmer_review.farmerabout, farmer = farmer_auth)
                if farmer_in_book:
                    farmer_review.book = farmer_in_book
            farmer_book = None
            if farmer_auth and farmer_auth != farmer_from:
                farmer_in_book = Farmerbook.objects.filter(farmeron =
farmer_from, farmer = farmer_auth)
                if farmer_in_book:
                    farmer_book = farmer_in_book
            return render(request, "farmrenter/oreviews.html",
{"reviews": farmer_reviews, "farmerbook": farmer_book, "farmer":
farmer_from })

# перегляд інформаційної картки сільськогосподарської тварини
фермера
def farm_animal_instance_info_view(request,
farmanimalinstance_id):

```

```

        if request.user.is_authenticated:
            try:
                farmer_auth = Farmer.objects.get(farmerdjango =
request.user)
            except Farmer.DoesNotExist:
                farmer_auth = None
        else:
            farmer_auth = None
        try:
            farm_animal_instance_found =
Farmanimalinstance.objects.select_related("farmanimal",
"farmanimal__farmanimalsort",
"farmanimal__farmanimalsort__farmanimaltype", "farmanimal__farm",
"farmanimal__farm__farmregion", "farmanimal__farm__farmer").get(id =
farmanimalinstance_id)
        except Farmanimalinstance.DoesNotExist:
            farm_animal_instance_found = None
        if not farm_animal_instance_found:
            return render(request, "farmrenter/error.html", {"mess":
"Про таку тварину нам нічого не відомо."})
        farmer_book = None
        if farmer_auth:
            farmer_in_book = Farmerbook.objects.filter(farmeron =
farm_animal_instance_found.farmanimal.farm.farmer, farmer =
farmer_auth)
            if farmer_in_book:
                farmer_book = farmer_in_book
            farm_animal_book = None
            if farmer_auth:
                farm_animal_in_book =
Farmanimalbook.objects.filter(farmanimal =
farm_animal_instance_found.farmanimal, farmer = farmer_auth)
                if farm_animal_in_book:
                    farm_animal_book = farm_animal_in_book
            farmanimal_reviews =
Farmanimalinstancereview.objects.select_related("farmer",
"farmer__farmerdjango").filter(farmanimalinstance__id =
farmanimalinstance_id)
            farmanimal_reviews = farmanimal_reviews.order_by("-
farmreviewtime")
            farmanimal_reviews = farmanimal_reviews[:5]
            for farmanimal_review in farmanimal_reviews:
                farmer_in_book = None

```

```

        if farmer_auth:
            farmer_in_book = Farmerbook.objects.filter(farmeron
= farmanimal_review.farmer, farmer = farmer_auth)
            if farmer_in_book:
                farmanimal_review.book = farmer_in_book
            else:
                farmanimal_review.book = None
        else:
            farmanimal_review.book = None
    if farm_animal_instance_found.farmanimal.farm.farmer ==
farmer_auth:
        import datetime
        current_date = datetime.datetime.now().date()
        farmrental_animals =
Farmrentalanimal.objects.select_related("farmrental").filter(farmanima
linstances__in = farm_animal_instance_found,
farmrental__farmrentalstart__lte = current_date,
farmrental__farmrentalend__gte =
current_date).values("farmrental__id").distinct()
        farmrental_statuses =
Farmrentalstatus.objects.filter(farmrental__in = farmrental_animals,
farmrentalstatusset = "3")
        if farmrental_statuses:
            farm_animal_instance_found.infarm = False
        else:
            farm_animal_instance_found.infarm = True
    return render(request, "farmrenter/farminst.html",
{"farmerbook": farmer_book, "animalbook": farm_animal_book, "animal":
farm_animal_instance_found, "reviews": farmanimal_reviews })

#перегляд інформаційної картки фермера
def farmer_info_view(request, farmer_login = None):
    if request.user.is_authenticated:
        try:
            farmer_auth = Farmer.objects.get(farmerdjango =
request.user)
        except Farmer.DoesNotExist:
            farmer_auth = None
    else:
        farmer_auth = None
    try:

```

```

        farmer_to =
Farmer.objects.select_related("farmerdjango").get(farmerdjango__userna
me = farmer_login)
        except Farmer.DoesNotExist:
            farmer_to = None
        if not farmer_to:
            return render(request, "farmrenter/error.html", {"mess":
"Про такого фермера нам нічого не відомо."})
        farmer_reviews =
Farmerreview.objects.select_related("farmer",
"farmer__farmerdjango").filter(farmerabout = farmer_to)
        farmer_reviews = farmer_reviews.order_by("-farmreviewtime")
        farmer_reviews = farmer_reviews[:5]
        for farmer_review in farmer_reviews:
            farmer_review.book = None
            if farmer_auth:
                farmer_in_book = Farmerbook.objects.filter(farmeron
= farmer_review.farmer, farmer = farmer_auth)
                if farmer_in_book:
                    farmer_review.book = farmer_in_book
            farmer_book = None
            if farmer_auth:
                farmer_in_book = Farmerbook.objects.filter(farmeron =
farmer_to, farmer = farmer_auth)
                if farmer_in_book:
                    farmer_book = farmer_in_book
        farms =
Farm.objects.select_related("farmregion").filter(farmer = farmer_t)
        return render(request, "farmrenter/farmer.html", {"reviews":
farmer_reviews, "farmerbook": farmer_book, "farmer": farmer_to,
"farms": farms })

```

ДОДАТОК В
Слайди презентації

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»
КАФЕДРА ПРОГРАМНИХ ЗАСОБІВ**

**ТЕМА ДИПЛОМНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА:
«ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ОРЕНДИ
СІЛЬСЬКОГОСПОДАРСЬКИХ ТВАРИН
FARM ANIMAL RENTAL SOFTWARE»**

Виконавець:

Дмитро ВІННІЧЕНКО

Керівник: к.т.н., доцент

Олександр СТЕПАНЕНКО

2025

Рисунок В.1 – Слайд перший

ОБ'ЄКТ, ПРЕДМЕТ, МЕТА РОБОТИ

- Об'єкт роботи – процеси оренди сільськогосподарських тварин.
- Предмет роботи – програмне забезпечення оренди сільськогосподарських тварин.
- Мета роботи – розробити програмне забезпечення оренди сільськогосподарських тварин для надання можливості користувачам знаходити варіанти тимчасового володіння тваринами в господарських завданнях.

2

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

ПОРІВНЯННЯ ПРОГРАМ ОРЕНДИ СІЛЬСЬКОГОСПОДАРСЬКИХ ТВАРИН

Характеристика	Rentthechicken	Benaysanimals	Farmrenter
Оформлення замовлення на оренду сільсько-господарських тварин	Без конкретних дат	Без конкретних дат	З датами та кількістю
Множинне представлення фермерів-власників тварин	Немає	Немає	Надається
Визначення обсягів отриманих продуктів від оренди	Немає	Немає	Надається
Керування власними замовленнями оренди сільськогосподарських тварин	Немає	Немає	Надається

3

Рисунок В.3 – Порівняння програм оренди сільськогосподарських тварин

ПОРІВНЯННЯ ЗАСОБІВ РОЗРОБКИ ПРОГРАМИ ОРЕНДИ СІЛЬСЬКОГОСПОДАРСЬКИХ ТВАРИН

Характеристика	CherryPy	Django
Структура застосунку	Визначається розробником	MVT
Застосування	Невеликі прості проекти	Великі, середні проекти, керовані даними
Об'єктно-реляційне відображення	Немає, можна додати	Одразу після встановлення готове до використання
Рушій для створення шаблонів вебсторінок	Немає, можна додати	Одразу після встановлення готовий до використання

4

Рисунок В.4 – Порівняння засобів розробки програми оренди сільськогосподарських тварин

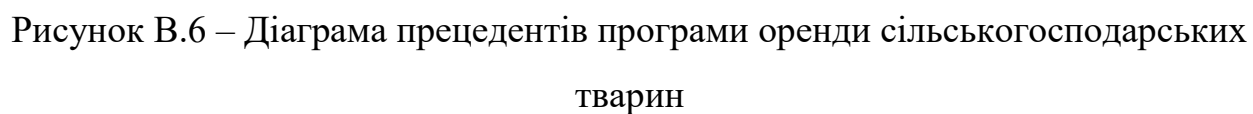
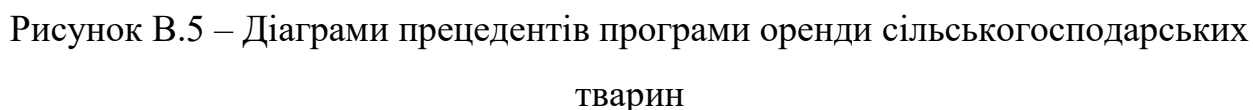


СХЕМА БАЗИ ДАНИХ ПРОГРАМИ ОРЕНДИ СІЛЬСЬКОГОСПОДАРСЬКИХ ТВАРИН

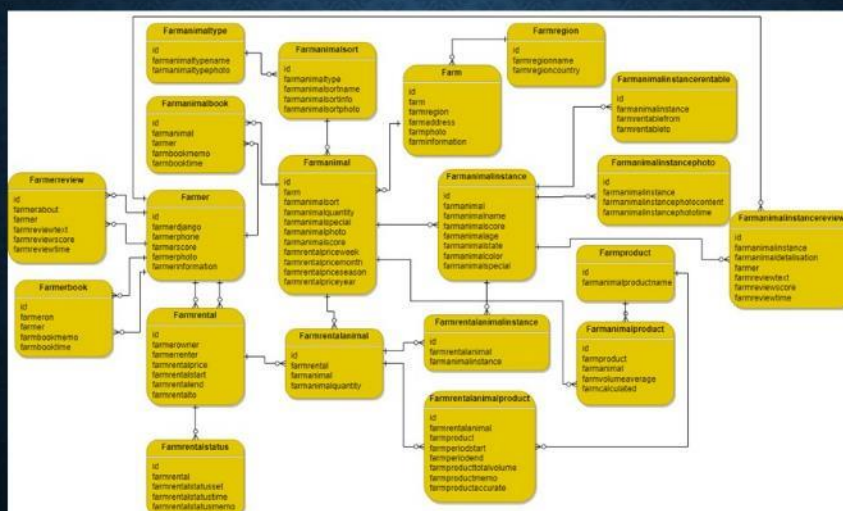


Рисунок В.7 – Схема бази даних програми оренди сільськогосподарських тварин

СХЕМА ФУНКЦІОНУВАННЯ ПРОГРАМИ ОРЕНДИ СІЛЬСЬКОГОСПОДАРСЬКИХ ТВАРИН

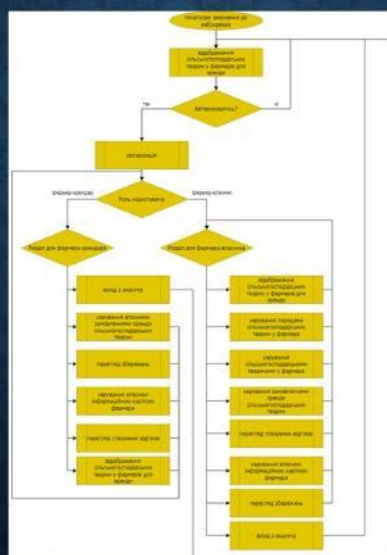


Рисунок В.8 – Схема функціонування програми оренди
сільськогосподарських тварин

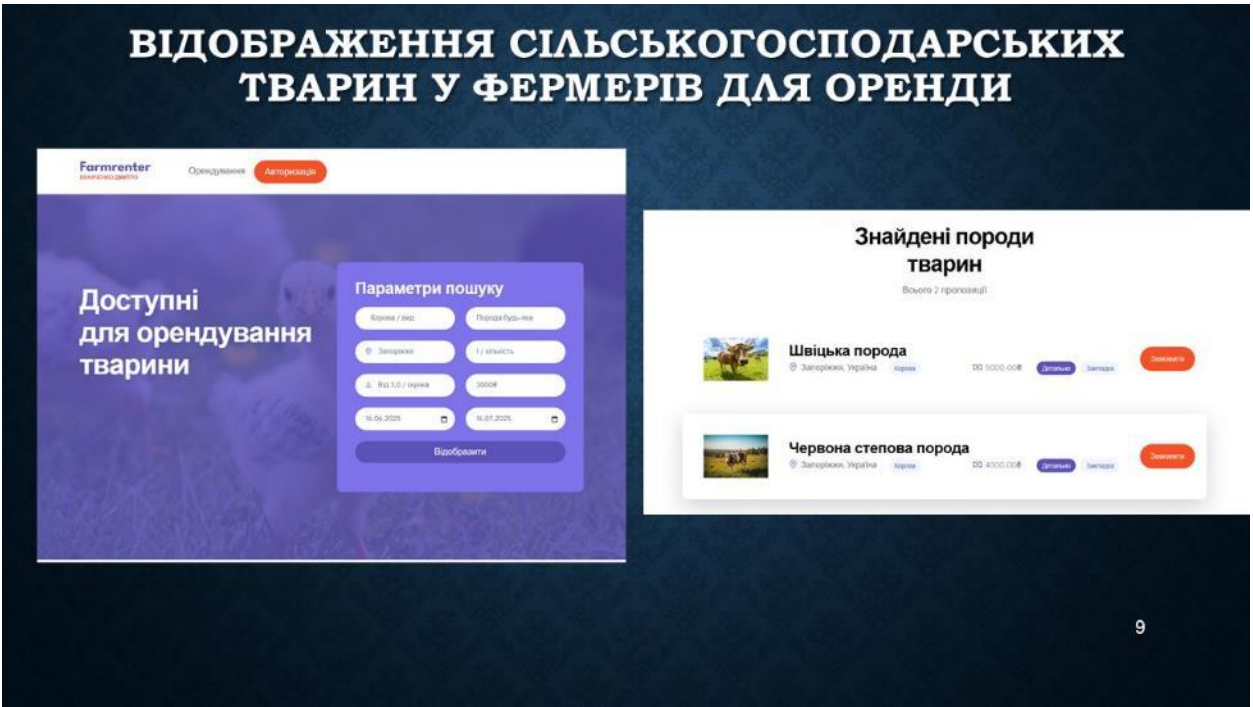


Рисунок В.9 – Відображення сільськогосподарських тварин у фермерів для оренди

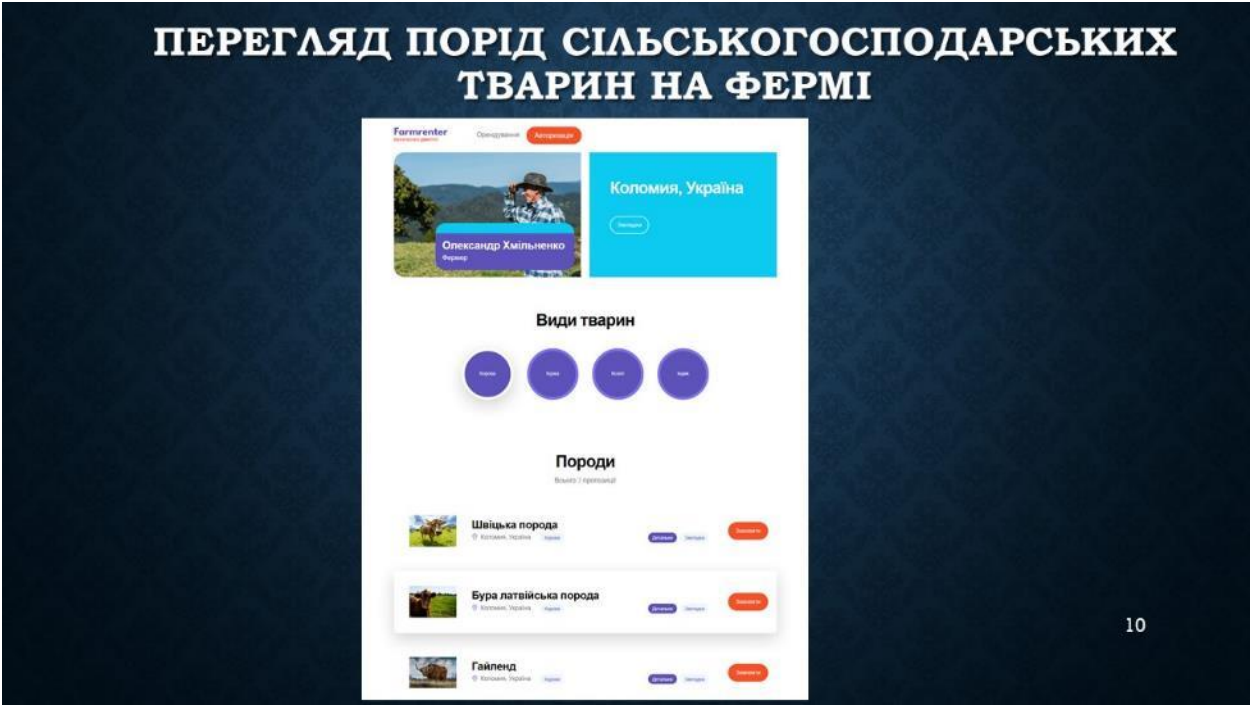


Рисунок В.10 – Перегляд порід сільськогосподарських тварин на фермі

ПЕРЕГЛЯД ЗАМОВЛЕНЬ ОРЕНДИ СІЛЬСЬКОГОСПОДАРСЬКИХ ТВАРИН

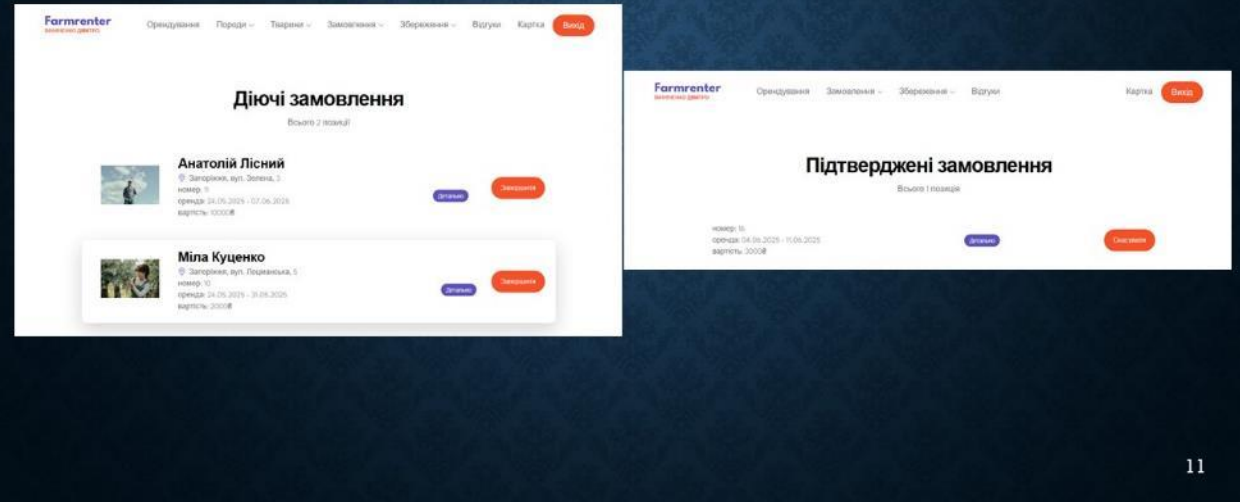


Рисунок В.11 – Перегляд замовлень оренди сільськогосподарських тварин

ВИСНОВКИ

- Програма оренди сільськогосподарських тварин створена на основі аналізу існуючого ринку, загальних тенденцій та тенденцій, які виникають під час коливань на ринку, а також аналізу існуючих програмних засобів для розв'язання цієї проблеми.
- Реалізовані програмні одиниці застосунку Django для забезпечення всіх функцій програми за структурою шаблону Model-View-Template. Також для реалізації програми використано мову Python і систему керування базою даних MySQL.
- Створена програма призначена для оренди сільськогосподарських тварин у власників на певний період часу, надання власниками своїх сільськогосподарських тварин для оренди.

Рисунок В.12 – Висновки