Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для керування вирощуванням сільськогосподарських культур рослин
(тема)

Виконав:
здобувач 4 року навчання
групи ПЗПІз-21-1

Катерина ЯРЕМЧУК
(Власне ім'я, ПРІЗВИЩЕ)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. кафедри ПІ Андрій БАБІЙ
(посада, Власне ім'я, ПРІЗВИЩЕ)

Допускається до захисту
Зав. кафедри _____ Кирило СМЕЛЯКОВ
(підпис) (Власне ім'я, ПРІЗВИЩЕ)

2025 р.

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Яремчук Катерині Олександрівні
(прізвище, ім'я, по батькові)

1. Тема роботи Програмна система для керування вирощуванням сільськогосподарських культур рослин
Затверджена наказом по університету від 05.05.2025р. № 74 Стз
2. Термін подання студентом роботи до екзаменаційної комісії 13.06.2025
3. Вихідні дані до роботи Розробити програмну систему для керування вирощуванням культур у сільському господарстві, що забезпечує автоматизацію агротехнічних процесів. Для реалізації використовувати React.js для фронтенду, FastAPI на Python для бекенду та MongoDB для бази даних.
4. Перелік питань, що потрібно опрацювати в роботі
Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проєктування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	06.05.2025	*виконано*
2	Створення специфікації ПЗ	10.05.2025	*виконано*
3	Проектування ПЗ	12.05.2025	*виконано*
4	Розробка ПЗ	13.05.2025	*виконано*
5	Тестування ПЗ	18.05.2025	*виконано*
6	Оформлення пояснювальної записки	20.05.2025	*виконано*
7	Підготовка презентації та доповіді	29.05.2025	*виконано*
8	Попередній захист	02.06.2025	*виконано*
9	Нормоконтроль, рецензування	04.06.2025	*виконано*
10	Здача роботи у електронний архів	09.06.2025	*виконано*
11	Допуск до захисту у зав. кафедри	12.06.2025	*виконано*

Дата видачі завдання « 5 » « травня » 2025р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. кафедри ПІ Андрій БАБІЙ
(підпис) (посада, Власне ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра, 81 стор., 32 рис., 1 табл., 21 джерел.

АГРОІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ, АГРОТЕХНІЧНЕ ПЛАНУВАННЯ, АВТОМАТИЗАЦІЯ АГРОПРОЦЕСІВ, ПРОГРАМНА СИСТЕМА, СІЛЬСЬКЕ ГОСПОДАРСТВО, УПРАВЛІННЯ ВИРОЩУВАННЯМ КУЛЬТУР, MONGODB, PYTHON, REACT.JS, SMART FARMING.

Об’єкт розробки – програмна система для автоматизованого управління вирощуванням сільськогосподарських культур.

Мета розробки – створення інтелектуальної програмної системи, що дозволяє оптимізувати агротехнічні заходи, автоматично формувати календар догляду, підбирати оптимальні культури та забезпечувати ефективне використання ресурсів з урахуванням агроекологічних умов.

Метод рішення – середовище розробки Visual Studio Code, фреймворк React.js для фронтенду, мова програмування Python для бекенду, MongoDB для бази даних, а також інструменти тестування (Selenium, Postman, юніт-тести).

У результаті розробки створено інтелектуальну програмну систему, яка забезпечує автоматизоване планування догляду за культурами, оптимізацію їх розміщення на полі, формування рекомендацій щодо підбору культур, а також зручний інтерфейс для взаємодії з користувачем. Система дозволяє підвищити ефективність сільськогосподарських процесів, зменшити витрати ресурсів і підвищити врожайність, відповідаючи сучасним вимогам сталого агровиробництва.

ABSTRACT

AGRO-INFORMATION TECHNOLOGIES, AGROTECHNICAL PLANNING, AGRICULTURAL PROCESS AUTOMATION, SOFTWARE SYSTEM, AGRICULTURE, CROP CULTIVATION MANAGEMENT, MONGODB, PYTHON, REACT.JS, SMART FARMING.

The object of development is a software system for automated management of crop cultivation.

The purpose of the work is to create an intelligent software system that allows optimizing agrotechnical measures, automatically generating a maintenance schedule, selecting optimal crops, and ensuring efficient use of resources, taking into account agroecological conditions.

Solution method – Visual Studio Code development environment, the React.js framework for the front end, the Python programming language for the back end, MongoDB for the database, as well as testing tools (Selenium, Postman, unit tests).

As a result of the development, an intelligent software system that provides automated crop care planning, optimization of their placement in the field, recommendations for crop selection, and a user-friendly interface. The system allows you to increase the efficiency of agricultural processes, reduce resource consumption, and increase yields, meeting the modern requirements of sustainable agricultural production.

Я, Яремчук Катерина Олександрівна, студент гр. ПЗПІз-21-1, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система для керування вирощуванням сільськогосподарських культур рослин», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлена з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API – Application Programming Interface;

CSV – Comma-Separated Values;

IoT – Internet of Things;

JWT – JSON Web Token;

MongoDB – Mongo Database;

PDF – Portable Document Format;

REST – Representational State Transfer;

UI – User Interface;

UML – Unified Modeling Language;

UX – User Experience.

ВСТУП

Сільське господарство – це одна з найважливіших і найстаріших галузей людської діяльності, яка безпосередньо впливає на соціально-економічний розвиток країн та забезпечує харчову безпеку населення. У сучасних умовах стрімкого зростання чисельності населення на планеті, урбанізації, зміни клімату та обмеженості природних ресурсів, перед аграрним сектором постають нові виклики, що вимагають застосування інноваційних технологій та ефективних методів управління.

Застосування традиційних методів ведення сільського господарства часто є недостатньо ефективним, оскільки вони базуються на досвіді та інтуїції фермерів, що призводить до нераціонального використання земельних ресурсів, добрив, води та інших матеріалів. Це, в свою чергу, спричиняє зниження врожайності, підвищення собівартості продукції та негативний вплив на навколишнє середовище через перенасичення ґрунтів хімікатами та нераціональне використання водних ресурсів.

В останні десятиліття спостерігається стрімкий розвиток інформаційних технологій і засобів автоматизації, що відкривають нові можливості для аграрної галузі. Зокрема, системи автоматизованого управління вирощуванням культур дозволяють підвищити продуктивність, точність агротехнічних заходів, а також забезпечити комплексний підхід до планування, моніторингу і контролю агровиробництва [1]. Використання таких систем допомагає інтегрувати різноманітні дані про стан ґрунтів, кліматичні умови, біологічні особливості культур та економічні показники, що дозволяє приймати більш обґрунтовані та ефективні рішення.

Однією з актуальних задач є створення програмних систем, які не лише виконують моніторинг і аналіз, а й пропонують оптимальні агротехнічні заходи, адаптовані до конкретних умов господарства. Це особливо важливо для малих і середніх фермерських підприємств, де відсутні можливості для застосування дорогого обладнання та кваліфікованого персоналу, а також для великих

агрохолдингів, що прагнуть мінімізувати витрати і підвищити ефективність виробництва.

Враховуючи все це, розробка програмної системи, яка автоматично формує календар догляду за рослинами, оптимізує їх розміщення на полі, враховуючи характеристики ґрунту, освітлення та потреби культур, а також рекомендує оптимальний набір рослин для конкретних екологічних умов, є надзвичайно актуальною і перспективною. Впровадження таких рішень сприятиме підвищенню продуктивності, зниженню витрат, покращенню якості продукції та забезпеченню сталого розвитку аграрного сектору.

Метою даної роботи є розробка інтелектуальної програмної системи для керування процесом вирощування сільськогосподарських культур, що дозволить автоматизувати та оптимізувати агротехнічні операції, підвищити ефективність використання ресурсів та адаптувати виробничі процеси під конкретні умови ґрунту, клімату і біологічних особливостей культур. Програма має забезпечити комплексний підхід, що включає автоматичне планування догляду за рослинами, оптимізацію їх розміщення на полях і вибір найбільш придатних культур з урахуванням введених параметрів середовища.

Реалізація цієї мети сприятиме підвищенню врожайності, зниженню витрат на полив, добрива і захист рослин, а також зменшенню негативного впливу на навколишнє середовище. Крім того, система має бути зручною у використанні для фермерів з різним рівнем підготовки, що робить її універсальним інструментом для широкого кола користувачів.

Для досягнення поставленої мети необхідно розв’язати низку конкретних завдань, серед яких:

- аналіз існуючих методів і засобів управління сільськогосподарськими процесами: вивчення сучасних підходів до автоматизації агротехнічних операцій, аналіз переваг і недоліків існуючих систем, формулювання вимог до нової програмної системи;

- проектування архітектури програмної системи: розробка загальної структури програмного продукту, визначення його основних модулів, баз даних та механізмів взаємодії компонентів;
- розробка алгоритмів формування календаря догляду: автоматизація планування агротехнічних заходів – поливу, внесення добрив, обробки від шкідників і хвороб – на основі біологічних характеристик культур;
- створення механізмів оптимізації розміщення рослин: розробка алгоритмів, що враховують розмір культур, їхні потреби у освітленні, тип ґрунту та сумісність сусідніх рослин для підвищення продуктивності і раціонального використання площі;
- реалізація модулю рекомендацій для вибору культур: автоматичний підбір оптимального набору рослин з урахуванням кліматичних умов, типу ґрунту, доступності води та інших параметрів середовища;
- розробка зручного інтерфейсу користувача: забезпечення простоти і зручності введення даних, відображення рекомендацій і нагадувань;
- тестування і оцінка ефективності системи: проведення експериментальних досліджень на основі реальних або змодельованих агротехнічних сценаріїв, аналіз результатів і внесення покращень.

Розроблена програмна система може бути застосована у різноманітних сегментах аграрного бізнесу: від невеликих фермерських господарств до великих агрохолдингів, а також у наукових і освітніх установах, що займаються агротехнологіями і екологічним менеджментом. Вона стане корисним інструментом для агрономів, агротехніків і фермерів, які прагнуть підвищити продуктивність своєї роботи і зменшити втрати.

Система сприятиме раціоналізації сільськогосподарських процесів, що відповідає сучасним тенденціям цифрової трансформації агросектору та концепції «розумного» фермерства (Smart Farming). Впровадження таких інновацій дозволяє підвищити конкурентоспроможність сільськогосподарської продукції, зберегти природні ресурси та зменшити вплив на навколишнє середовище.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Сільське господарство – одна з ключових галузей економіки будь-якої країни, яка забезпечує продовольчу безпеку, формує значну частку національного ВВП, забезпечує зайнятість населення та є стратегічною з точки зору екологічного і соціального розвитку [2]. В умовах глобальних викликів, таких як зміна клімату, зменшення площ родючих земель, зростання населення та потреби в продовольстві, підвищується необхідність модернізації сільського господарства шляхом впровадження сучасних інформаційних технологій.

Одним із найважливіших аспектів діяльності аграрного сектора є процес вирощування сільськогосподарських культур, що охоплює цілий цикл дій – від підготовки ґрунту до збору врожаю. Цей цикл включає багатоетапну систему взаємозалежних операцій: вибір культури, визначення відповідного типу ґрунту, аналіз кліматичних умов, планування сівозміни, підбір агротехнічних заходів (обробка ґрунту, полив, підживлення, захист від шкідників), спостереження за розвитком рослин і, врешті, збирання врожаю. Кожен із цих етапів вимагає відповідального підходу та прийняття рішень, які впливають на продуктивність та ефективність аграрного виробництва.

У традиційній моделі господарювання більшість рішень приймаються фермерами на основі власного досвіду, сезонних спостережень або інтуїції. Проте такий підхід не завжди є ефективним, особливо в умовах нестабільного клімату, нестачі інформації, великої площі угідь чи обмеженого доступу до агрономічної підтримки. Саме тому все більшої актуальності набуває застосування інформаційних систем і програмних рішень, які здатні оптимізувати процеси вирощування культур, підвищити врожайність, зменшити втрати та мінімізувати людський фактор.

Процес вирощування сільськогосподарських культур є складною технологічною системою, що охоплює кілька ключових компонентів:

- ґрунтово-кліматичні умови – тип ґрунту (суглинковий, піщаний, глинистий), його кислотність, рівень вологості, наявність мінеральних елементів, температура повітря і ґрунту, кількість сонячного світла, рівень опадів та вітрове навантаження;
- агротехнічні вимоги конкретної культури – це оптимальні умови, за яких культура має найвищу продуктивність. Наприклад, томати потребують теплого клімату, великої кількості сонячного світла та регулярного поливу, тоді як картопля є більш стійкою до перепадів температур;
- фази розвитку культури – кожна рослина проходить кілька етапів розвитку: проростання, формування листя, бутонізація, цвітіння, плодоношення, дозрівання. На кожному з цих етапів потреби рослини змінюються;
- захист рослин – боротьба зі шкідниками, хворобами, бур'янами є критично важливою для збереження врожаю. Це включає обприскування, механічну обробку та біологічні методи;
- моніторинг стану рослин – спостереження за змінами зовнішнього вигляду, кольору листя, наявністю плям, відставанням у рості дозволяє вчасно виявити проблеми.

Усі ці аспекти потребують збору, зберігання, обробки й аналізу великої кількості інформації, що створює передумови для застосування програмної системи як інструменту автоматизації і прийняття рішень.

Світовий тренд переходу до Smart Farming або розумного фермерства ґрунтується на інтеграції сучасних технологій в агровиробництво. До таких технологій належать:

- геоінформаційні системи (GIS) – для створення карт полів, зонування, визначення родючості;
- інтернет речей (IoT) – для встановлення сенсорів вологості, температури або кислотності ґрунту;
- аналіз великих даних (Big Data) – для аналізу погодних трендів, захворювань рослин, історії врожаїв;

- штучний інтелект і машинне навчання – для прогнозування врожайності, виявлення хвороб, формування рекомендацій;
- мобільні застосунки та веб-платформи – для зручного доступу до даних, агрокалендарів, рекомендацій.

На цьому тлі програмна система, що дозволяє фермеру отримати персоналізовані рекомендації щодо вирощування обраних культур з урахуванням локальних умов – є логічним і затребуваним кроком.

Незважаючи на наявність технічного прогресу, малим та середнім фермерським господарствам часто не вистачає знань, ресурсів або часу для якісного аналізу агроумов. Серед основних проблем, з якими стикаються аграрії:

- невідповідність культури умовам середовища, що призводить до низької врожайності або загибелі рослин;
- недотримання агротехнічних термінів – посів, полив, обробка виконуються надто рано чи надто пізно;
- занадто загальні або комерційно орієнтовані поради з інтернету, які не враховують локальної специфіки;
- відсутність доступних інструментів для планування та контролю агроциклу.

У відповідь на вищезазначені проблеми виникає потреба у створенні універсальної, доступної, адаптивної програмної системи, яка б дозволяла:

- автоматично підбирати культури для вирощування, виходячи з параметрів середовища (типу ґрунту, клімату, вологості);
- формувати персоналізований календар догляду, який враховує фазу росту рослини, локальні погодні умови та рекомендації агрономії;
- вести облік висаджених культур, їхнього стану, результатів попередніх сезонів;
- забезпечувати зручний, зрозумілий і функціональний інтерфейс як для досвідчених аграріїв, так і для початківців.

Таким чином, запропонована програмна система виступає посередником між сучасними цифровими технологіями і практичними потребами фермерів, сприяючи оптимізації агропроцесів, підвищенню врожайності та зниженню витрат.

Сучасне сільське господарство активно впроваджує інноваційні цифрові технології, що дозволяють підвищити ефективність вирощування культур, знизити витрати ресурсів і збільшити врожайність. На ринку існує великий спектр програмних продуктів, які допомагають аграріям планувати роботи на полях, контролювати стан культур і приймати обґрунтовані рішення. Проте, кожна з цих систем має свої сильні й слабкі сторони, що робить актуальним створення нової, більш адаптованої до конкретних потреб користувачів – зокрема, невеликих і середніх фермерських господарств України.

Cropio – це хмарна агроплатформа, розроблена для моніторингу сільськогосподарських угідь у реальному часі (див. рис. 1.1). Вона використовує супутникові знімки, погодні дані та інші джерела, щоб відслідковувати стан рослинності, стан ґрунту та інші критично важливі параметри.

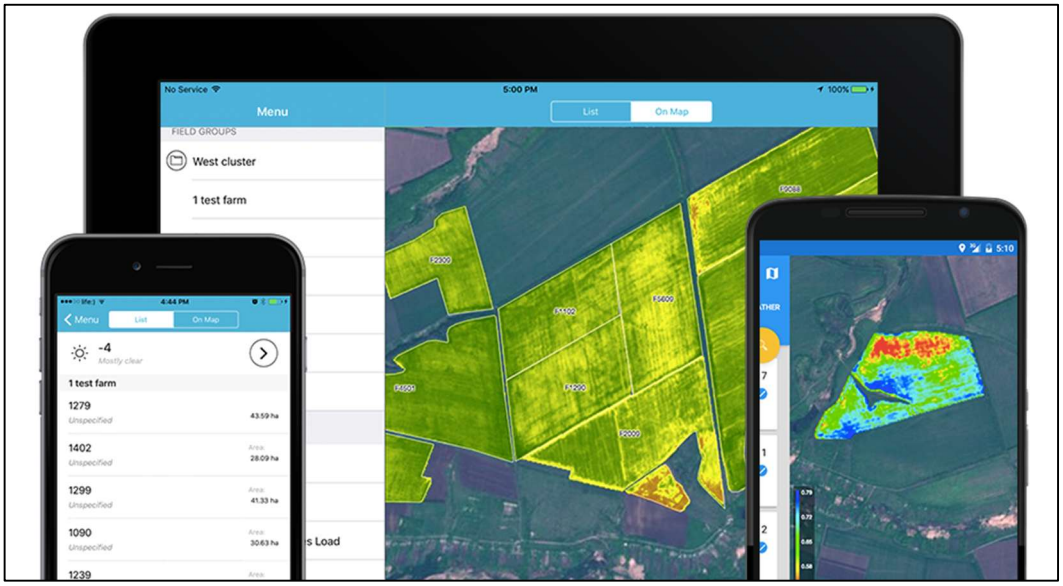


Рисунок 1.1 – Додаток Cropio (за даними [3])

Головні функції:

– відображення карт полів з кольоровим позначенням зон із різним рівнем вегетації;
– прогнозування врожайності на основі історичних та поточних даних;

– інтеграція з метеостанціями для отримання локальної інформації про погоду;
– автоматичне сповіщення про ризики, такі як посуха чи надмірна вологість.

Переваги Cropio:

– висока точність моніторингу завдяки використанню супутникових знімків високої роздільної здатності;
– інтуїтивний інтерфейс із зручним візуальним відображенням;
– потужні інструменти для аналізу тенденцій розвитку культур.

Обмеження:

– основна аудиторія – великі агрохолдинги з великими площами оброблюваних земель;
– вартість підписки може бути суттєвою для малих і середніх господарств;
– система не пропонує глибокої персоналізації під окремі культури з урахуванням локальних умов;
– обмежена можливість інтеграції з іншими локальними сервісами, зокрема українськими.

AgriTask позиціонує себе як комплексне рішення для управління агровиробництвом, що охоплює весь цикл: від планування польових робіт до контролю якості й логістики (див. рис. 1.2). Платформа інтегрується з різними пристроями – датчиками ґрунту, дронами, метеостанціями.

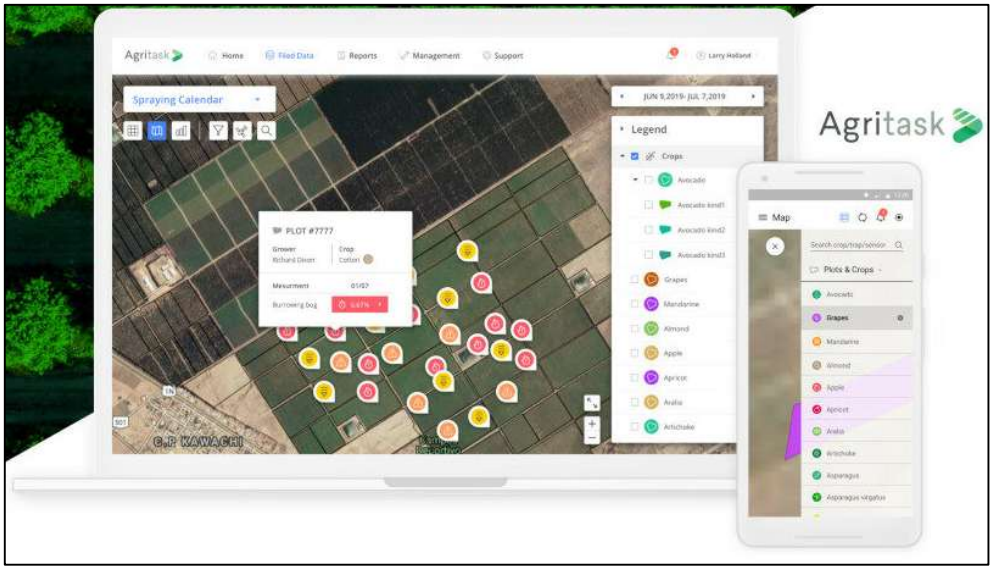


Рисунок 1.2 – Додаток AgriTask (за даними [4])

Ключові можливості:

– планування робіт із деталізацією для кожного поля;
– облік ресурсів і техніки;
– автоматизовані звіти для агрономів і менеджерів;
– використання машинного навчання для прогнозування проблем і рекомендацій.

Переваги:

– гнучкість налаштувань під специфіку господарства;
– широкий спектр інтеграцій з апаратним забезпеченням;
– підтримка складних агрономічних моделей.

Недоліки:

– потребує високої кваліфікації користувача для повноцінної роботи;
– імовірно висока вартість ліцензій;
– складність інтерфейсу може бути перешкодою для дрібних фермерів.

OneSoil – це відносно проста і доступна платформа для моніторингу полів із використанням супутникових даних (див. рис. 1.3). Вона пропонує безкоштовний базовий функціонал для аграріїв.

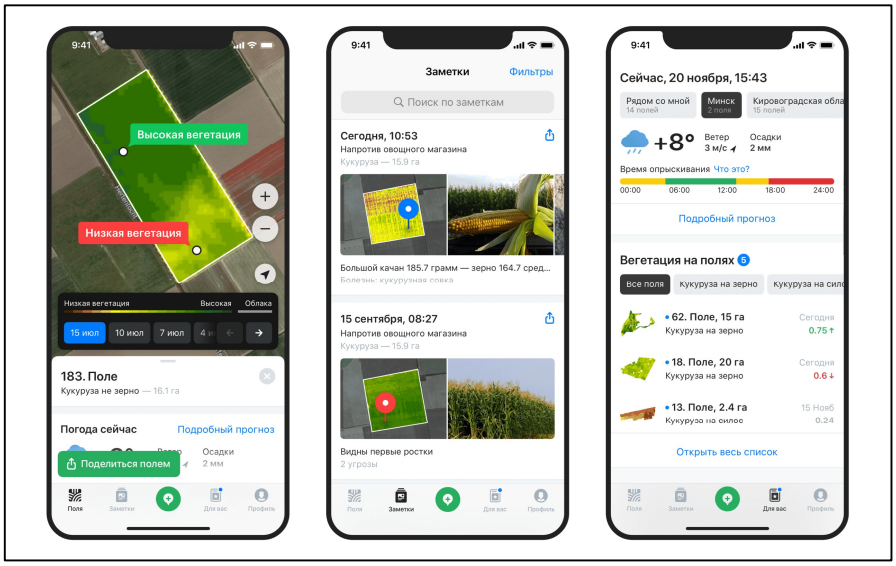


Рисунок 1.3 – Додаток OneSoil (за даними [5])

Основний функціонал:

– відображення карт полів із індексом вегетації NDVI;

– створення нотаток і позначок для конкретних зон;

– підрахунок площі посівів;

– оцінка здоров’я рослинності у різні періоди.

Переваги:

– безкоштовна або недорога;

– простий інтерфейс, що дозволяє швидко почати роботу;

– мобільний застосунок, що дає змогу працювати в полі.

Недоліки:

– відсутність глибоких аналітичних модулів;

– не підтримує автоматичне створення календарів догляду;

– не дає рекомендацій щодо добрив, поливу або сівозміни;

– немає можливості врахувати особливості ґрунту і клімату.

Plantix – мобільний додаток, який спеціалізується на виявленні хвороб рослин за допомогою аналізу фотографій (див. рис. 1.4). Він використовує технології штучного інтелекту для розпізнавання симптомів захворювань, шкідників і дефіциту поживних речовин.

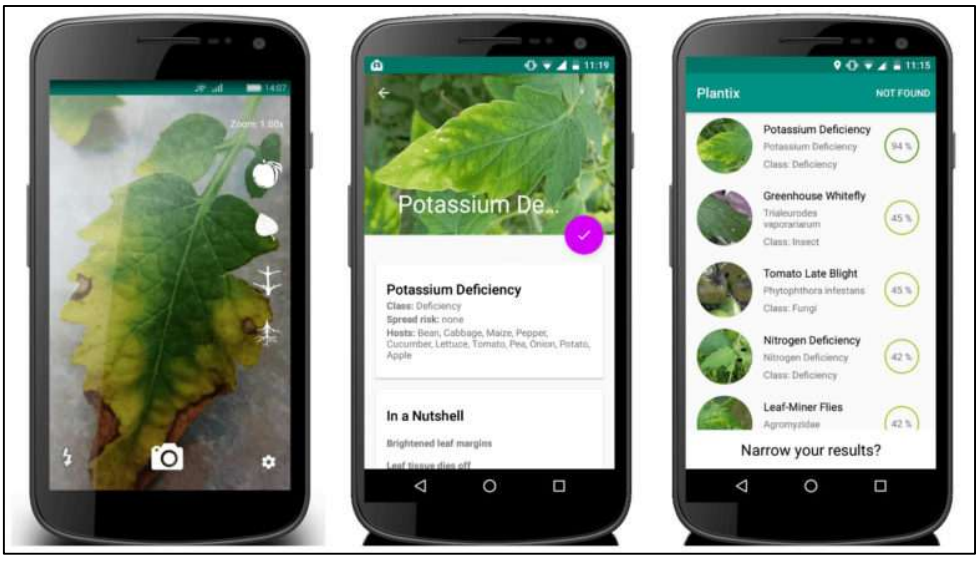


Рисунок 1.4 – Додаток Plantix (за даними [6])

Функції:

– фотографування рослин і миттєвий аналіз;

– виявлення хвороб і шкідників;

– надання рекомендацій щодо лікування і профілактики;

– соціальна мережа для обміну досвідом серед фермерів.

Переваги:

– легкість у використанні для будь-якого рівня користувача;

– велика база знань про хвороби та шкідників;

– можливість роботи офлайн після завантаження бази.

Недоліки:

– не охоплює планування агротехнічних заходів;

– не підтримує довгострокове планування сівозміни чи догляду;

– відсутність модулів для оптимізації розміщення культур або управління ресурсами.

Agroop – це система для збору та аналізу агроданих з використанням сенсорів і датчиків, що допомагає оптимізувати витрати на воду, добрива та інші ресурси (див. рис. 1.5). Підходить для моніторингу в режимі реального часу.

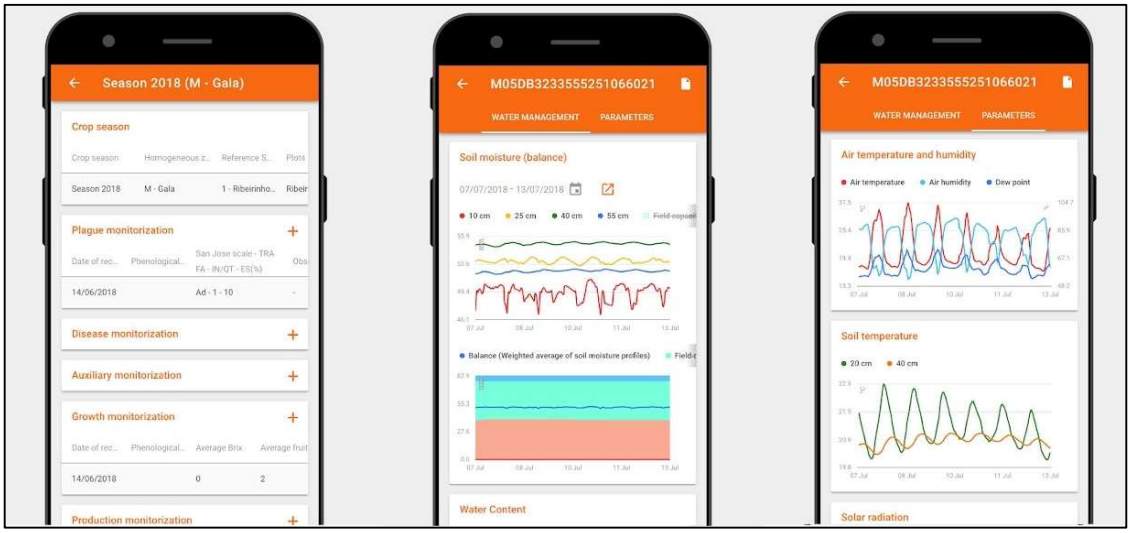


Рисунок 1.5 – Додаток Agroop (за даними [7])

Особливості:

– збір даних із сенсорів ґрунту, кліматичних датчиків;

– віддалений моніторинг стану полів;

– автоматичні оповіщення про ризики;

– планування та аналіз агровиробництва.

Переваги:

- висока деталізація даних про стан культури і ґрунту;
- підвищення ефективності використання ресурсів;
- зручна аналітика у вигляді графіків та звітів;

Недоліки:

- потребує інвестицій у обладнання;
- не всі функції підходять для господарств із низьким рівнем технічного оснащення;
- немає глибоких порад щодо агротехнічних заходів на основі введених параметрів.

Наявність на ринку різноманітних інструментів свідчить про актуальність і попит на цифрові технології у сільському господарстві. Однак жоден із розглянутих аналогів не дає вичерпного рішення, яке б забезпечило повний цикл керування вирощуванням культур – від вибору оптимального набору рослин відповідно до умов ґрунту та клімату, через планування догляду і розміщення, до нагадувань про агротехнічні заходи.

Тож розробка нової програмної системи керування вирощуванням культур у сільському господарстві, яка враховує ці потреби та є доступною для широкого кола користувачів, має вагомі перспективи. Вона сприятиме підвищенню ефективності виробництва, оптимізації використання ресурсів і, як наслідок, зростанню економічної стабільності фермерських господарств.

1.2 Виявлення та вирішення проблем

У сучасному сільському господарстві ефективне управління процесами вирощування сільськогосподарських культур має вирішальне значення для підвищення продуктивності та економічної вигоди. Однак на практиці існує низка суттєвих проблем, які стримують розвиток агровиробництва та знижують його ефективність. Однією з головних проблем є низький рівень автоматизації агропроцесів, особливо у малих та середніх господарствах, де більшість операцій ведеться вручну або на основі суб’єктивних рішень. Це призводить до

нераціонального планування посівів, неправильного використання добрив та засобів захисту рослин, що в кінцевому результаті знижує врожайність та збільшує витрати. Вирішення цієї проблеми передбачає впровадження програмних систем, які автоматизують планування, моніторинг та контроль всіх етапів вирощування культур, забезпечують централізоване зберігання та аналіз даних, що дозволяє приймати обґрунтовані рішення.

Іншою суттєвою проблемою є відсутність системного моніторингу стану рослин та ґрунту. Часто агровиробники не мають оперативної інформації про рівень вологості ґрунту, стан живлення рослин, наявність хвороб або шкідників, що унеможливлює швидке реагування на негативні фактори. Внаслідок цього виникають втрати врожаю і зниження якості продукції. Сучасні технології дозволяють вирішити цю проблему шляхом інтеграції програмного забезпечення з датчиками, що вимірюють параметри середовища, а також із супутниковими системами дистанційного зондування. Це дає можливість автоматично збирати дані, аналізувати їх та формувати рекомендації або попередження для користувача в режимі реального часу.

Ще однією проблемою є складність ведення агрономічної документації та звітності. Відсутність єдиного електронного реєстру призводить до розпорошеності інформації, втрати даних, значних витрат часу на підготовку звітів для сертифікацій, контролюючих органів та внутрішнього обліку. Використання програмної системи дозволяє автоматизувати документообіг, ведення журналів обробітків, облік використаних матеріалів і ресурсів, а також формувати звіти у стандартизованому вигляді, що значно спрощує звітність і забезпечує прозорість.

Недостатнє планування та прогнозування є ще однією перешкодою на шляху підвищення ефективності сільськогосподарського виробництва. Без інструментів, які враховують історичні дані, погодні умови та особливості ґрунтів, фермери змушені працювати "наосліп", що підвищує ризики невдалих посівів, втрат врожаю та непродуктивних витрат. Впровадження аналітичних модулів у програмне забезпечення, які базуються на сучасних алгоритмах обробки даних і машинному

навчанні, дає змогу робити точні прогнози строків посіву, строків внесення добрив та захисту рослин, а також очікуваної врожайності.

Великим ризиком у сільському господарстві залишається людський фактор, який призводить до помилок у дозуванні препаратів, пропуску важливих агротехнічних заходів, неправильного розподілу ресурсів. Система, що автоматично нагадує про необхідні операції, контролює дотримання технологій, а також веде журнал дій з позначенням відповідального користувача, мінімізує цей ризик і підвищує дисципліну роботи в господарстві.

Окрім цього, відсутність контролю за економічною ефективністю агроопераций ускладнює оцінку рентабельності посівів і призводить до неефективного використання ресурсів. Автоматизовані модулі фінансової аналітики дають змогу вести облік усіх витрат і доходів, аналізувати собівартість продукції, порівнювати ефективність різних культур і заходів, що дозволяє аграріям приймати стратегічні рішення на основі конкретних даних.

При масштабуванні виробництва керувати усіма процесами стає ще складніше через збільшення площ, кількості робітників та техніки. Відсутність централізованої системи контролю знижує оперативність і якість управління. Запровадження багатокористувацьких платформ із різними рівнями доступу, можливістю віддаленого моніторингу та контролю робіт сприяє підвищенню продуктивності, дозволяє координувати роботу всіх підрозділів господарства і зменшити витрати часу на комунікацію.

Таким чином, виявлені проблеми в сільському господарстві мають комплексний характер і пов’язані як із технологічними, так і організаційними аспектами. Розроблена програмна система для керування вирощуванням культур спрямована на їх комплексне вирішення. Вона автоматизує основні процеси планування, моніторингу, обліку та аналізу, забезпечує оперативний доступ до даних, підтримує прийняття обґрунтованих рішень та знижує вплив людського фактора. Впровадження такої системи дозволить підвищити врожайність, зменшити витрати, покращити якість продукції і значно підвищити загальну ефективність сільськогосподарського виробництва.

1.3 Постановка задачі

Сучасне сільське господарство, зокрема вирощування культур, перебуває між традиційними методами та потребою впровадження цифрових технологій. Кліматичні зміни, зростання витрат і нестача кваліфікованих кадрів змушують фермерів шукати ефективні засоби для планування й оптимізації агропроцесів. Проте багато господарств і досі не використовують автоматизовані системи, що призводить до втрат урожайності, нераціонального використання ресурсів і зниження прибутків.

З огляду на це, виникає необхідність у створенні програмної системи, здатної комплексно вирішувати завдання управління вирощуванням культур з урахуванням сучасних агрономічних підходів, екологічних вимог та доступності для кінцевого користувача.

Метою створення програмної системи є автоматизація процесів планування, моніторингу та аналізу вирощування сільськогосподарських культур, що дозволить підвищити ефективність господарювання, зменшити втрати та покращити якість прийняття рішень в агровиробництві.

Для досягнення цієї мети необхідно реалізувати низку завдань, які охоплюють як розробку окремих функціональних модулів, так і забезпечення їх інтеграції в єдину систему:

- аналіз аграрного середовища: дослідити типові сценарії ведення агровиробництва, виявити потреби та проблеми фермерів, оцінити існуючі підходи до вирощування різних культур з урахуванням типів ґрунту, кліматичних умов і наявних ресурсів;
- побудова бази знань про рослини: сформувати структуровану базу даних, що містить відомості про різноманітні культури, включаючи їхні біологічні особливості, потреби в освітленні, вологозабезпеченні, частоті поливу, добривах, оптимальних строках висіву та збирання врожаю;
- розробка модуля автоматичного планування календаря догляду: створити механізм, що, на основі введених параметрів культур та умов середовища,

автоматично генерує персоналізований календар агротехнічних заходів із прив’язкою до дат (полив, підживлення, обробка від шкідників) та реалізує систему нагадувань;

- оптимізація просторового розміщення культур: реалізувати інструмент, який враховує потреби культур у світлі, просторі та складі ґрунту, і пропонує найбільш раціональне розміщення рослин на ділянці або полі;
- аналіз умов середовища та рекомендації з підбору культур: надати користувачеві можливість введення параметрів (тип ґрунту, клімат, доступ до води) для того, щоб система на основі алгоритмів прийняття рішень пропонувала перелік культур, що найкраще підходять для вирощування в конкретних умовах;
- інтерфейс користувача: створити інтуїтивно зрозумілий веб-інтерфейс, який дозволяє легко додавати культури, переглядати розклад агрооперацій, отримувати сповіщення, редагувати налаштування;
- гнучкість і масштабованість: забезпечити можливість розширення системи новими модулями (наприклад, інтеграція з датчиками вологості ґрунту, погодними API, модулями прогнозування врожайності), адаптації під різні розміри господарств і типи користувачів;
- автоматизований моніторинг виконання: реалізувати механізми фіксації фактів виконання або пропуску агротехнічних заходів, щоб система могла аналізувати динаміку догляду, адаптувати рекомендації та попереджати про ризики;
- забезпечення зберігання історичних даних: реалізувати журнал подій та операцій для подальшого аналізу ефективності агропрактик, навчання моделей прогнозування та підвищення точності рекомендацій.

Загалом, постановка задачі полягає у створенні програмного комплексу – цифрового помічника фермера/агронома, що автоматизує рутинні завдання, покращує управлінські рішення, зменшує залежність від людського фактору та підвищує продуктивність. Основна увага приділяється потребам користувача, зручності, надійності та адаптивності системи до умов агросектору.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Проєктування та розробка програмної системи є складним багаторівневим процесом, який вимагає чіткого визначення функціональних і нефункціональних вимог ще на початкових етапах. Формування вимог дозволяє встановити межі системи, структурувати її функціональність, визначити сценарії використання, а також забезпечити узгодженість між очікуваннями користувачів та реалізацією програмного продукту. Враховуючи специфіку предметної галузі – сільське господарство, – особливу увагу слід приділити простоті використання, адаптивності до умов реального середовища, масштабованості, надійності та інтеграції з зовнішніми джерелами інформації (клімат, погода, дані ґрунту).

Програмна система має забезпечувати комплексне управління процесами вирощування сільськогосподарських культур. Основними цільовими користувачами є фермери, агрономи, працівники сільськогосподарських підприємств, а також приватні особи, які займаються землеробством на малих площах. Програма повинна працювати в середовищі веб-доступу (браузер).

Система повинна надавати користувачам можливість планувати та оптимізувати агротехнічні процеси з урахуванням конкретних характеристик культур і умов середовища, надавати персоналізовані рекомендації, відстежувати виконання завдань, автоматично формувати календарі догляду та генерувати повідомлення про необхідні дії.

Функціональні вимоги – це перелік функцій, які система має виконувати для задоволення потреб користувача. Вони охоплюють основну логіку поведінки системи, доступні операції, дії користувача, інтерфейсну взаємодію та обробку даних.

Реєстрація та автентифікація користувачів:

- система повинна забезпечувати реєстрацію нових користувачів із прив’язкою до електронної пошти або номера телефону;
- після реєстрації має бути доступна авторизація за логіном/паролем;
- підтримка скидання пароля;

– ролі користувачів: базова (фермер, агроном), адміністратор (повний доступ).

Керування земельними ділянками:

– додавання, редагування та видалення земельних ділянок користувачем;
– вказівка площі, геолокації, типу ґрунту, джерел води, освітленості.

Додавання культур:

– користувач має змогу додати культуру вручну або обрати з вбудованої бази знань;
– для кожної культури вказуються такі параметри: назва, тривалість вегетації, частота поливу, періодичність внесення добрив, оптимальні умови вирощування (ґрунт, світло, температура, вологість);
– дані використовуються системою для формування персоналізованого плану догляду.

Автоматичне створення календаря догляду:

– система автоматично формує календар необхідних агротехнічних заходів (полив, добрива, прополювання) на основі параметрів кожної культури;
– користувач отримує повідомлення/нагадування про необхідні дії через веб-інтерфейс або мобільні сповіщення;
– є можливість ручного редагування календаря за бажанням користувача.

Оптимальне розміщення рослин на полі:

– на основі вказаних параметрів (розмір культури, потреби в освітленні, тип ґрунту) система автоматично пропонує оптимальне розміщення рослин на заданій площі;
– підтримується відображення розміщення у вигляді сітки або схеми;
– враховується сумісність культур та чергування сівозмін.

Підбір культур за параметрами середовища:

– користувач вводить або завантажує параметри середовища (тип ґрунту, кліматична зона, середня температура, наявність зрошення);

– система аналізує введені дані та пропонує список культур, які найбільше підходять для вирощування в таких умовах;
– є можливість перегляду докладних характеристик кожної рекомендованої культури.

Моніторинг виконання агрооперацій:

– користувач відмічає виконання запланованих дій;
– у разі невиконання – система перебудовує графік;
– дані використовуються для формування аналітики.

Управління базою знань культур:

– адміністратор має змогу оновлювати список культур, імпортувати нові записи, редагувати або видаляти існуючі;
– користувачі можуть переглядати базу та додавати власні сорти (опціонально).

Генерація звітів:

– автоматичне створення звітів про виконання робіт, ефективність та відповідність плану;
– експорт у формати PDF/CSV.

Нефункціональні вимоги – це вимоги, які визначають якість роботи системи, а не її конкретну поведінку. Вони описують обмеження та характеристики, що впливають на ефективність, надійність, масштабованість, безпеку, зручність використання та інші важливі аспекти програмного забезпечення. Іншими словами, нефункціональні вимоги відповідають на питання "як повинна працювати система", а не "що вона повинна робити".

Зручність та простота інтерфейсу:

– інтерфейс має бути інтуїтивно зрозумілим, не перевантаженим деталями;
– підтримка української мови інтерфейсу;
– доступність для користувачів без технічної підготовки.

Продуктивність:

- завантаження сторінок має здійснюватися не більше ніж за 3 секунди при стандартному з’єднанні;
- час відповіді на запити користувача не має перевищувати 1 секунди для більшості операцій.

Масштабованість:

- система має підтримувати розширення функціоналу без необхідності повної переробки архітектури;
- можливість підключення додаткових модулів (аналіз ґрунту, прогнозування врожайності, погодні API).

Надійність і доступність:

- мінімальна кількість помилок при експлуатації;
- захист даних користувача: регулярне збереження, резервне копіювання;
- безвідмовна робота не менше ніж 99% часу протягом року.

Безпека:

- шифрування персональних даних користувача;
- захист від SQL-ін’єкцій, XSS, CSRF;
- розмежування прав доступу.

Сумісність:

- підтримка сучасних браузерів (Chrome, Firefox, Edge);
- адаптація під мобільні пристрої (респонсивний дизайн).

Економічність:

- оптимальне використання обчислювальних ресурсів;
- можливість розгортання на недорогому хостингу.

Обмеження – це частина нефункціональних вимог, яка визначає умови, межі або правила, яких необхідно дотримуватися під час проєктування, розробки, розгортання або експлуатації програмної системи. Вони часто обумовлені зовнішніми чинниками: ресурсними обмеженнями, нормативно-правовими актами, стандартами якості, вимогами до сумісності, доступними технологіями. Встановлені обмеження визначають рамки, у яких розробляється система, і

допомагають уникнути технічних, логістичних чи експлуатаційних труднощів на етапі реалізації.

Нижче наведено обмеження для програмної системи:

- обмеження щодо продуктивності: система має працювати коректно при одночасному використанні до 10 активних користувачів. Масштабування на більшу кількість користувачів буде розглядатись після аналізу навантаження у реальних умовах експлуатації;
- використання лише відкритих технологій і фреймворків: система має бути побудована з використанням технологій з відкритим вихідним кодом, без залежності від комерційних ліцензій, щоб уникнути додаткових витрат та правових обмежень;
- немає офлайн-режиму: робота системи можлива лише за наявності стабільного підключення до Інтернету, оскільки доступ до бази даних, синхронізація з хмарними сервісами та оновлення вимагають постійного з’єднання;
- обмеження по обсягу збережених даних: у базовій конфігурації система не повинна зберігати понад 100 МБ мультимедійних файлів (зображення полів, фото культур) без очищення архіву або хмарної інтеграції;
- відсутність повноцінного мобільного застосунку: на першому етапі реалізується лише адаптивна веб-версія. Мобільний застосунок (на Android/iOS) може бути створений окремим етапом у разі високого попиту.

Отже, сформульовані вимоги охоплюють усі ключові аспекти програмної системи для керування вирощуванням культур. Чітке дотримання цих вимог у процесі розробки дозволить створити ефективний, зручний та адаптивний інструмент для сучасного аграрного користувача.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

У процесі розробки програмної системи для керування вирощуванням сільськогосподарських культур надзвичайно важливою є попередня формалізація логіки, структури та взаємодії компонентів. Одним із найбільш ефективних засобів такого моделювання є мова уніфікованого моделювання (UML), яка дозволяє візуалізувати архітектуру, поведінку та розгортання програмного забезпечення [8]. У рамках цієї роботи UML було використано для створення низки діаграм, які відображають як функціональну поведінку системи, так і її структурну та фізичну організацію. Це дозволило отримати чітке уявлення про взаємозв’язки між компонентами, їхню відповідальність, а також про способи обробки запитів користувача та зберігання даних.

Діаграма прецедентів для програмної системи керування вирощуванням культур у сільському господарстві містить три основні ролі користувачів – фермер/агроном та адміністратор, кожна з яких має власний набір функціональних можливостей, які відображають ключові сценарії використання системи (див. рис. 3.1).

Для ролей фермера та агронома система реалізує комплекс функціональних прецедентів, що охоплюють весь цикл взаємодії з платформою. Спершу користувач має можливість здійснити реєстрацію, вказуючи електронну пошту або номер телефону для створення облікового запису. Реєстрація є початковою дією, необхідною для подальшого використання системи. Після реєстрації фермер проходить процедуру авторизації, вводячи логін та пароль, що забезпечує безпечний доступ до персональних даних і функцій платформи. Важливо, що майже всі основні функції системи, окрім безпосередньої реєстрації та відновлення пароля, виконуються виключно після успішної авторизації, що підкреслює необхідність безперервної аутентифікації користувача.

У разі втрати пароля передбачена можливість його відновлення, що є логічним розширенням процесу авторизації і активується за потреби користувача. Після входу в систему фермер/агроном може додавати, редагувати або видаляти

інформацію про земельні ділянки, які він обробляє. Ця функція дозволяє вказати площу ділянки, тип ґрунту, джерела води та рівень освітленості, що є критично важливими параметрами для подальшого формування агротехнічних заходів.

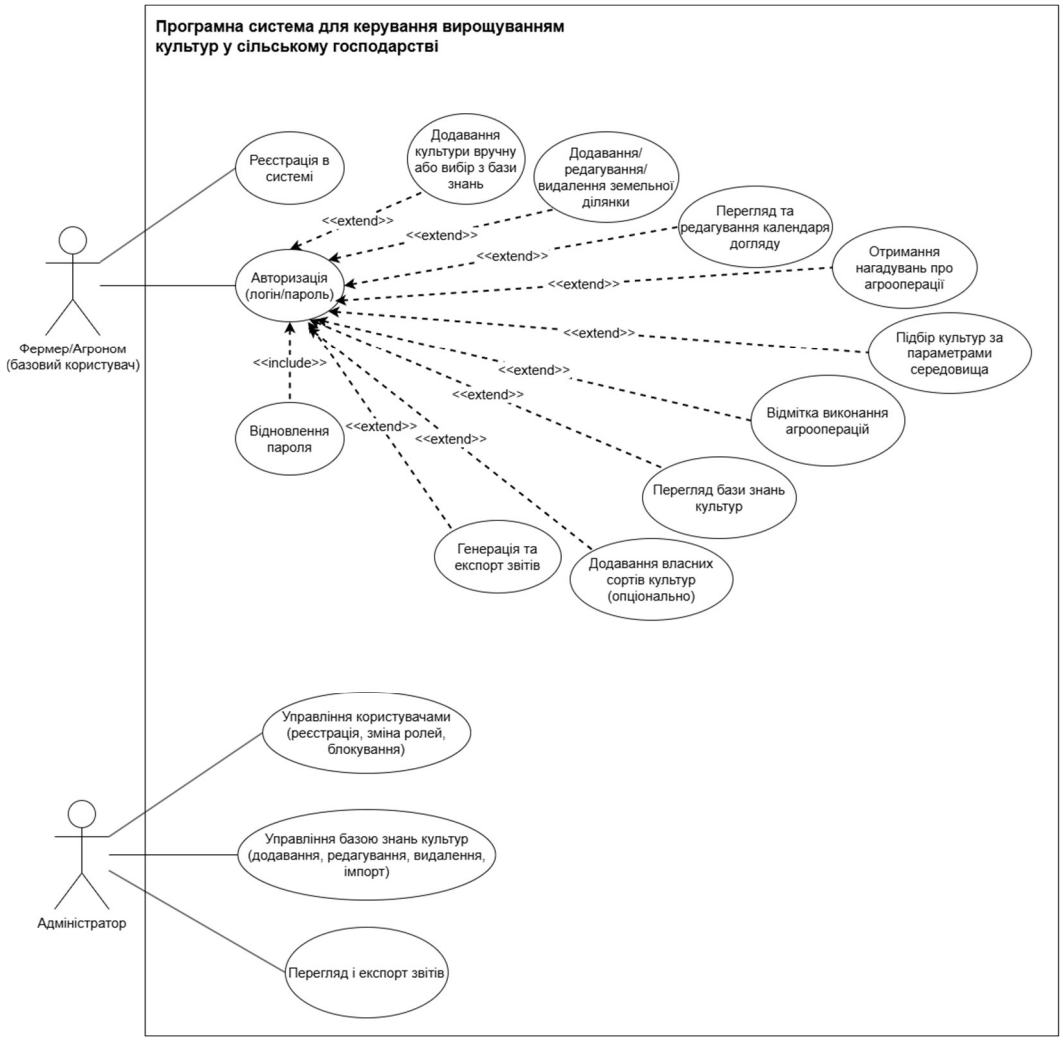


Рисунок 3.1 – Діаграма прецедентів (рисунок виконаний самостійно)

Для організації процесу вирощування культур фермер/агроном має можливість додавати нові культури вручну, вводячи необхідні параметри, або вибирати їх зі вбудованої бази знань, яка містить рекомендовані варіанти. Автоматично після додавання культури система формує персоналізований календар догляду, що включає полив, внесення добрив, прополювання та інші необхідні агротехнічні операції. Цей календар користувач може переглядати та при необхідності редагувати вручну, що забезпечує гнучкість у плануванні.

Ще однією важливою функцією є отримання нагадувань про заплановані агрооперації у вигляді сповіщень через веб-інтерфейс. Це дозволяє агроному не

пропустити важливі етапи догляду за рослинами. Також передбачено механізм підбору культур за параметрами середовища – користувач вводить тип ґрунту, кліматичну зону, середню температуру та наявність зрошення, а система аналізує ці дані й пропонує найбільш відповідні культури для вирощування у заданих умовах.

Для контролю за виконанням агрооперацій фермер/агроном має можливість відмічати фактичне виконання робіт. Якщо деякі дії не виконано у встановлений час, система перебудовує графік відповідно до актуальних обставин, що допомагає підтримувати актуальність плану і ефективність агровиробництва. Дані про виконання операцій використовуються для формування аналітичних звітів.

Користувач також може переглядати базу знань культур, яка містить детальні характеристики рослин, а за бажанням – додавати власні сорти, розширюючи інформаційну базу системи. Фермер/агроном має доступ до функції генерації та експорту звітів, які відображають стан виконання робіт, ефективність вирощування і відповідність плану. Звіти можна експортувати у формати PDF або CSV для подальшого аналізу або звітності.

Роль адміністратора в системі полягає у повному контролі та управлінні користувачами і базою знань. Адміністратор може виконувати реєстрацію нових користувачів, змінювати їх ролі, блокувати або розблоковувати доступ, що забезпечує безпеку та належне управління обліковими записами. Він також відповідає за підтримку та оновлення бази знань культур, включаючи додавання нових записів, редагування, видалення застарілої інформації та імпорт даних з зовнішніх джерел. Це гарантує актуальність і повноту інформації, доступної користувачам. Крім того, адміністратор має доступ до перегляду та експорту звітів, що дозволяє здійснювати моніторинг роботи системи на загальному рівні.

Агроном також є ключовим користувачем програмної системи, відповідальним за організацію процесів вирощування рослин у саду. Його функціонал зосереджений на використанні інструментів системи для прийняття обґрунтованих рішень щодо агротехнічного догляду за культурами. Агроном має доступ до модуля автоматичного створення календаря догляду за рослинами, де на

основі характеристик обраних культур (частота поливу, періодичність внесення добрив, захист від шкідників) система генерує персоналізований план догляду. Календар відображається у зручному форматі з наглядними нагадуваннями про майбутні дії, що дозволяє агроному вчасно реагувати на потреби рослин. Система також пропонує механізм оптимізації розміщення садових культур. Агроном вводить дані про доступну площу, освітлення, тип ґрунту, а також обирає культури з бази даних або додає нові вручну. Програма розраховує найефективнішу схему розташування рослин, враховуючи сумісність видів, потребу у просторі та освітленні, що сприяє підвищенню врожайності та раціональному використанню садової ділянки. Додатково агроном може використовувати інструмент підбору культур: вводячи параметри середовища (тип ґрунту, кліматичні умови, наявність системи поливу), він отримує рекомендації щодо найбільш придатних для вирощування рослин. Це дозволяє ефективно планувати садові роботи, зменшити ризики неврожаю та забезпечити екологічно збалансоване ведення господарства. Завдяки інтелектуальній підтримці системи, агроном може зосередитися на стратегічних агрономічних рішеннях, підвищуючи ефективність, якість та сталість садівництва.

Варто підкреслити, що для адміністратора всі функції доступні незалежно, без внутрішніх зв’язків <<include>> чи <<extend>>, оскільки його права передбачають повний та безперешкодний доступ до системи після авторизації.

Отже, діаграма прецедентів чітко структурує функціональність системи відповідно до ролей користувачів, відокремлюючи базовий функціонал фермера/агронома, що зосереджений на оперативному управлінні вирощуванням культур, та адміністративний контроль, що забезпечує підтримку та оновлення системи в цілому. Використання зв’язків <<include>> для авторизації у фермерських прецедентах підкреслює безпеку та логічну послідовність дій, а розширення <<extend>> для відновлення пароля додає гнучкість у роботі з аутентифікацією.

Загалом, дана діаграма прецедентів забезпечує чітке розуміння вимог до системи, допомагає організувати розробку програмного забезпечення і закладає

фундамент для подальшого моделювання детальних сценаріїв та проектування архітектури.

Діаграма послідовності відіграє ключову роль у візуалізації логіки взаємодії користувача з програмною системою на рівні обміну повідомленнями між основними компонентами (див. рис. 3.2).

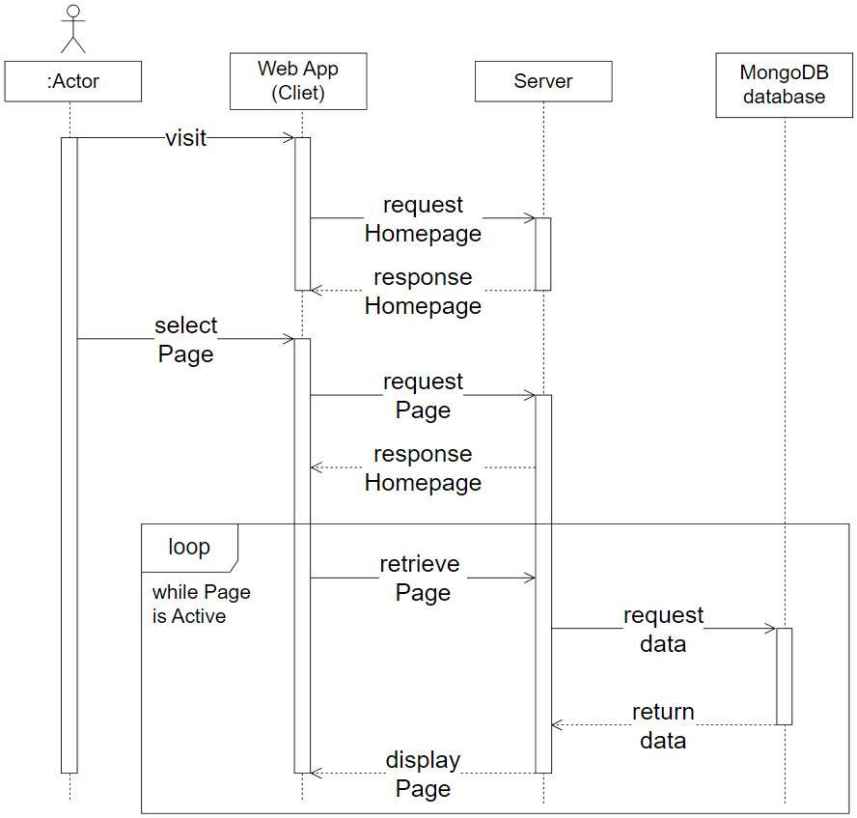


Рисунок 3.2 – Діаграма послідовності (рисунок виконаний самостійно)

Вона моделює хід подій у часі, починаючи з ініціації дії користувачем, наприклад, запиту на перегляд аграрних або метеоданих, і до завершення обробки цієї дії системою. Користувач (фермер, агроном) через графічний інтерфейс взаємодіє з клієнтською частиною застосунку, яка формує запит і передає його серверу. Сервер, у свою чергу, виконує обробку логіки, взаємодіє з базою даних MongoDB для читання чи оновлення необхідної інформації, після чого формує відповідь і надсилає її клієнтському інтерфейсу для відображення результату. Такий ланцюг дозволяє моделювати не лише окрему дію, а й загальний сценарій поведінки користувача в системі.

Діаграма пакетів структурує систему на логічно пов’язані модулі (пакети), кожен з яких відповідає за конкретний набір функціональностей (див. рис. 3.3).

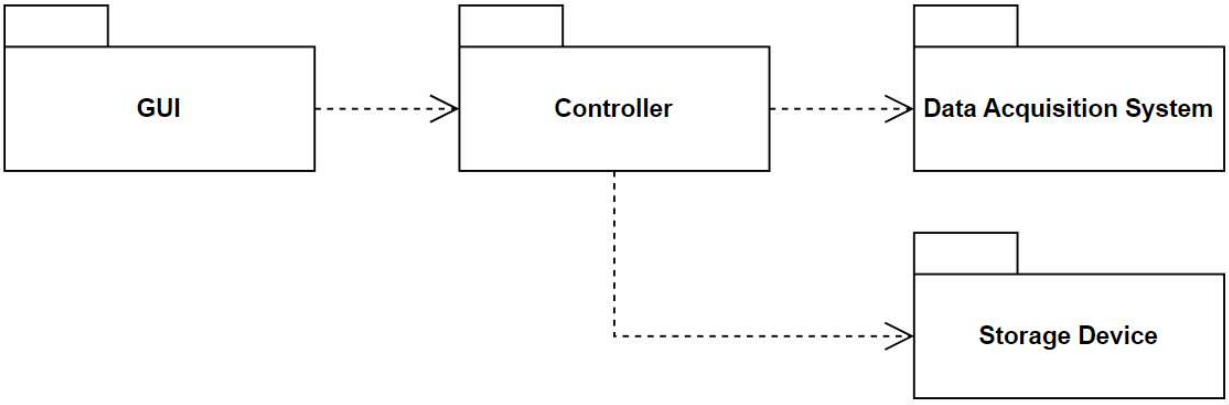


Рисунок 3.3 – Діаграма пакетів (рисунок виконаний самостійно)

У контексті аграрної інформаційної системи, що працює з великими масивами даних та потребує високої надійності, така модульність є критично важливою. Основними компонентами є:

- графічний інтерфейс користувача (GUI) відповідає за відображення інформації та прийом команд користувача. Тут реалізовано інтуїтивно зрозумілий візуальний доступ до аналітики, прогнозів, карт посівів та календаря агротехнічних робіт;
- контролерна частина – логіка, що забезпечує взаємозв’язок між інтерфейсом та бізнес-логікою. Вона приймає запити від GUI, формує відповідні дії та перенаправляє їх до необхідного модуля обробки;
- модуль збору та обробки даних (Data Acquisition System) відповідає за обробку, фільтрацію, валідацію та аналітику аграрних, кліматичних і супутникових даних. Включає механізми прогнозування врожайності, виявлення аномалій, оцінки стану ґрунту та моделювання ризиків;
- сховище даних на базі MongoDB – нереляційна база, яка дозволяє зберігати гнучкі структури даних у форматі JSON-документів. Це дає змогу оперативно оновлювати записи, ефективно працювати з геопросторовими даними та обробляти інформацію в режимі реального часу.

Такий поділ сприяє масштабованості, тобто можливості додавати нові функціональні блоки (наприклад, API для дронів або IoT-пристроїв) без необхідності перебудови існуючої логіки. Крім того, модульний підхід спрощує супровід і тестування системи, дозволяє проводити оновлення окремих блоків без зупинки всієї інфраструктури.

Діаграма розгортання ілюструє фізичну архітектуру програмної системи та її розміщення на реальному апаратному забезпеченні (див. рис. 3.4). Вона демонструє, як програмні компоненти розподілені між пристроями та серверами. Згідно з цією моделлю, користувач взаємодіє із застосунком через клієнтський пристрій (смартфон, планшет, ноутбук), підключений до мережі Інтернет.

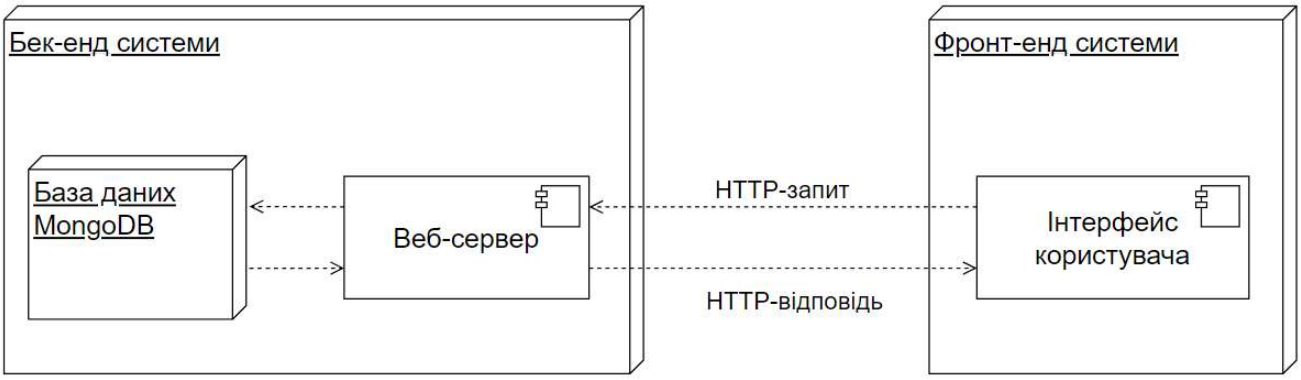


Рисунок 3.4 – Діаграма розгортання (рисунок виконаний самостійно)

Клієнтський застосунок надсилає HTTP(S)-запити на вебсервер, який розміщено або у хмарному середовищі (AWS, Azure, GCP), або локально на виділеному сервері [9]. Цей сервер виконує основну логіку обробки, звертаючись за необхідності до окремого сервера бази даних MongoDB. Така архітектура відповідає класичній клієнт-серверній моделі, що забезпечує стабільність, масштабованість та безпеку. У випадку аграрного сектору особливо важливими є показники доступності (враховуючи сезонність робіт), витривалості до пікових навантажень (у періоди збору чи прогнозування врожаю) і здатності до обробки великих обсягів геопросторової інформації, погодних шарів, аналітики з сенсорів. Додатково можливе використання кеш-серверів (Redis, Memcached) для прискорення доступу до популярних запитів і балансувальників навантаження для забезпечення безперебійної роботи при зростанні кількості користувачів.

3.2 Проєктування архітектури ПЗ

У процесі проєктування архітектури програмного забезпечення для системи керування вирощуванням культур у сільському господарстві було прийнято концептуальне рішення щодо використання трирівневої клієнт-серверної архітектури, яка є однією з найпоширеніших і найефективніших моделей організації програмних систем, що передбачають розділення функціональності на логічні рівні з метою досягнення масштабованості, зручності в супроводі та розвитку, а також ефективного управління складністю. Обрана архітектура забезпечує чітке розмежування обов’язків між рівнями, зокрема, клієнтським рівнем, рівнем проміжного програмного забезпечення (тобто сервером застосунків) та рівнем доступу до даних, що у сукупності дозволяє досягти високої надійності, безпеки та адаптивності до зміни бізнес-логіки системи.

Основна ідея трирівневої архітектури полягає в тому, що користувач взаємодіє з системою через інтерфейс, який реалізовано на фронтенді у вигляді вебдодатку, створеного з використанням сучасного JavaScript-фреймворку React. Цей інтерфейс відповідає за відображення усієї інформації, спрощене введення даних, здійснення реєстрації та автентифікації користувачів, створення і редагування об'єктів, а також за інтерактивну взаємодію з календарем, розміщенням культур і рекомендаціями. Кожна дія користувача через інтерфейс формує HTTP-запит, який надсилається на проміжний рівень, що реалізований у вигляді RESTful API за допомогою високопродуктивного Python-фреймворку FastAPI [10]. Саме цей рівень відповідає за всю бізнес-логіку: автентифікацію та авторизацію користувачів із використанням JWT-токенів, обробку даних про земельні ділянки та культури, динамічну побудову календаря агротехнічних операцій, аналіз параметрів середовища, генерацію рекомендацій щодо оптимального розміщення культур на полі з урахуванням сумісності та чергування сівозмін, ведення історії виконання агрооперацій, автоматичну перебудову графіків у разі затримок чи невиконання дій, а також формування звітів про продуктивність та ефективність. Окрім цього, на серверному рівні реалізовано механізми управління базою знань культур, що включає функції додавання нових

записів, редагування параметрів, імпорту базових культур із зовнішніх джерел і доступ до цієї інформації користувачами системи.

Третій рівень, рівень даних, реалізовано за допомогою документо-орієнтованої нереляційної системи управління базами даних MongoDB, яка обрана з огляду на її здатність працювати з гнучкими структурами даних та вкладеними документами, що є надзвичайно важливим для аграрної галузі, де об’єкти можуть мати нестандартні й різноманітні характеристики [11]. MongoDB дозволяє зберігати користувачів, земельні ділянки з географічною прив'язкою, інформацію про культури з усіма агротехнічними параметрами, події календаря, звіти та історії дій у вигляді окремих колекцій. Зв’язки між цими сутностями реалізовані на рівні вкладених об'єктів та посилань, що значно пришвидшує доступ до даних та спрощує аналітичну обробку. Таким чином, серверна частина програми виступає посередником між клієнтом і базою даних: обробляє запити, звертається до бази, здійснює обчислення, фільтрацію або трансформацію даних та повертає результат клієнтському інтерфейсу у зручному форматі JSON.

На рисунку 3.5 представлено структурну схему трирівневої клієнт-серверної архітектури, яка візуалізує логіку взаємодії компонентів системи.

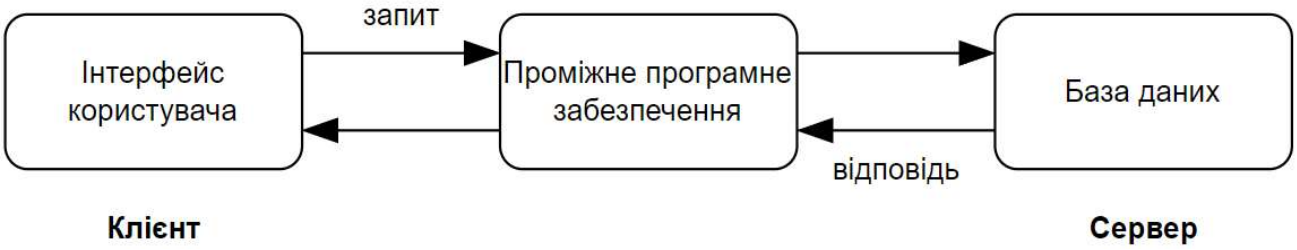


Рисунок 3.5 – Трирівнева клієнт-серверна архітектура системи (рисунок виконаний самостійно)

Зліва розташований клієнтський рівень, що представлений інтерфейсом користувача, через який надсилається запит. Цей запит передається до проміжного програмного забезпечення – тобто серверу застосунків, який реалізує всю логіку обробки, перевіряє права доступу, здійснює запити до бази даних, отримує звідти відповідь і повертає її користувачеві. Таким чином, інформаційна взаємодія здійснюється у вигляді послідовності «запит – обробка – відповідь». Ця модель

дозволяє забезпечити незалежність інтерфейсу від серверної логіки і бази даних, завдяки чому можлива гнучка модифікація кожного компонента без порушення роботи всієї системи.

Враховуючи специфіку предметної області, тобто аграрного сектора, обрана архітектура забезпечує масштабованість та гнучкість при обслуговуванні великої кількості користувачів, земельних ділянок і агрокультур з різними параметрами, а також дозволяє ефективно реалізувати механізми персоналізації, автоматизації планування, аналітики й рекомендацій. Таке архітектурне рішення також створює передумови для подальшої інтеграції з мобільними клієнтами, IoT-пристроями, датчиками вологості та температури або з відкритими метео-API, що у перспективі дозволить розширити функціональність системи до рівня повноцінної платформи для розумного землеробства. Таким чином, трирівнева архітектура забезпечує не лише відповідність функціональним вимогам, а й відповідає сучасним стандартам побудови інформаційних систем для агропромислового комплексу, поєднуючи надійність, продуктивність і перспективність розвитку.

3.3 Проєктування структури зберігання даних

Проєктування структури зберігання даних у MongoDB для програмної системи керування вирощуванням культур у сільському господарстві передбачає створення логічної, гнучкої та масштабованої моделі документів, що забезпечує ефективну взаємодію з великими обсягами інформації про користувачів, земельні ділянки, культури, вимірювання та агротехнічні рекомендації. Завдяки відсутності жорстких схем MongoDB дозволяє будувати дані в ієрархічній формі, використовуючи вкладені документи, що зменшує кількість запитів і підвищує продуктивність системи [12].

Для реалізації системи було спочатку створено класичну ER-діаграму (Entity-Relationship Diagram), яка слугує логічною моделлю даних (див. рис. 3.6).

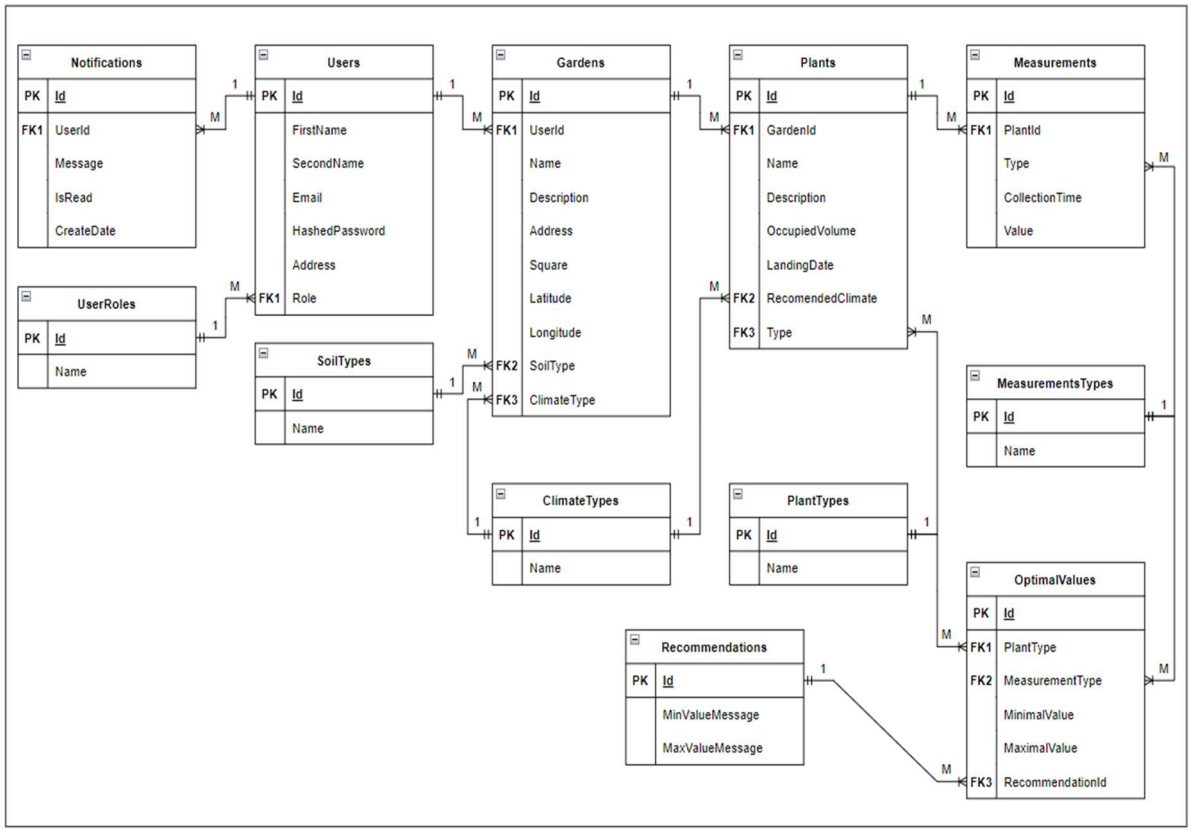


Рисунок 3.6 – Entity-Relationship Diagram структури даних системи управління агровирощуванням (рисунок виконаний самостійно)

Вона відображає ключові сутності (Users, Gardens, Plants, Measurements, Recommendations, Notifications, а також довідники SoilTypes, ClimateTypes, PlantTypes, MeasurementTypes, OptimalValues) із їх атрибутами та зв’язками між ними (один-до-багатьох, багато-до-одного). Наприклад, один користувач може мати кілька садів, кожен сад містить кілька рослин, для яких реєструються численні вимірювання, що аналізуються на відповідність оптимальним значенням і породжують відповідні рекомендації. Сповіщення пов’язані з користувачами для оперативного інформування про важливі події.

Однак, оскільки фізична реалізація проєкту передбачає використання MongoDB – документоорієнтованої NoSQL бази даних, класична реляційна модель трансформується у гнучку документну структуру з урахуванням особливостей MongoDB [13]. Замість таблиць і жорстких схем використовуються колекції (аналог таблиць) і документи формату BSON/JSON, що можуть містити вкладені об’єкти та масиви, що підвищує гнучкість і продуктивність [14].

Основні колекції системи:

- users: зберігає інформацію про зареєстрованих користувачів (ім’я, прізвище, email, захешований пароль, адреса, роль – фермер, агроном або адміністратор). Роль реалізується через внутрішній атрибут або вкладений об’єкт. Сповіщення (notifications) можуть зберігатися або як масив вкладених документів у користувача (за умови невеликого обсягу), або у вигляді окремої колекції з посиланням по userId;
- gardens: містить інформацію про земельні ділянки, включаючи посилання на користувача (userId), назву, опис, адресу, площу, геокоординати, тип ґрунту і кліматичний тип. Поля типу ґрунту та клімату можуть бути представлені як рядки або як посилання на відповідні довідникові колекції soilTypes і climateTypes;
- plants: зберігає дані про культури на конкретних садах (посилання на gardenId, назва, опис, об’єм, дата висадки, рекомендований клімат і тип рослини). Вимірювання (measurements) можуть бути вкладеними документами всередині рослини (якщо їх кількість незначна), або зберігатися в окремій колекції з посиланнями на рослину. Типи рослин і вимірювань зберігаються в довідниках plantTypes та measurementTypes;
- measurements: містять інформацію про тип вимірювання, дату збору та значення параметра. В залежності від обсягу даних, вимірювання можуть бути вкладеними у документи рослин або зберігатися окремо;
- recommendations: документи з текстовими рекомендаціями, що генеруються при відхиленнях параметрів від норми. Пов’язані з колекцією optimalValues, де зберігаються допустимі діапазони для кожного типу рослин і вимірювань;
- optimalValues: ключова колекція для аналізу відповідності умов вирощування заданим нормам, що дозволяє автоматично визначати необхідність попереджень (наприклад, надлишок вологості чи нестача світла) і формувати відповідні рекомендації;

- notifications: зберігає сповіщення для користувачів із посиланням на userId, текстом повідомлення, статусом прочитання та датою створення, що організує систему нагадувань про агротехнічні дії;
- довідникові колекції (soilTypes, climateTypes, plantTypes, measurementTypes) служать для централізованого управління нормативними значеннями і типами, хоча у деяких випадках можливе їх денормалізоване зберігання у вигляді вбудованих списків у коді застосунку, що прискорює читання даних.

Проєктування структури зберігання враховує баланс між нормалізацією та денормалізацією, а також між вкладеністю і посиланнями. Вкладені документи використовуються там, де це покращує продуктивність і зменшує кількість звернень до бази (наприклад, вимірювання всередині рослин). Посилання застосовуються для великих, часто оновлюваних сутностей (користувачі, рекомендації), щоб уникнути дублювання великих обсягів даних.

Гнучка схема MongoDB дозволяє легко додавати нові поля без необхідності редизайну всієї бази (наприклад, додаткові параметри саду чи рослини). Денормалізація деяких полів (назви типів ґрунтів або клімату) мінімізує необхідність складних операцій з’єднання, що відсутні у MongoDB.

Отже, розроблена модель зберігання даних забезпечує логічну відповідність між фізичним розташуванням даних та бізнес-процесами агросистеми, дозволяючи швидко отримувати інформацію про стан культур, надавати персоналізовані рекомендації, адаптувати догляд за рослинами та своєчасно інформувати користувача про критичні ситуації. Використання MongoDB гарантує масштабованість, гнучкість структури та високу швидкодію, що робить систему придатною для реального застосування в аграрній сфері.

3.4 Приклади найцікавіших алгоритмів та методів

У програмній системі для керування вирощуванням культур реалізовано низку інтелектуальних алгоритмів, які автоматизують ключові агротехнічні процеси, знижують ризики помилок та підвищують ефективність управління

рослинництвом. Однією з найцікавіших функціональних можливостей є автоматичне створення календаря догляду за рослинами, що базується на аналізі їхніх характеристик (див. рис. 3.7).

```
def generate_agro_calendar(plant_data):
    calendar = []
    start_date = datetime.fromisoformat(plant_data['planting_date'])
    days = plant_data['vegetation_duration']
    for day in range(0, days):
        current_date = start_date + timedelta(days=day)
        if day % plant_data['watering_freq'] == 0:
            calendar.append({"date": current_date.isoformat(), "task": "Полив"})
        if day % plant_data['fertilizer_freq'] == 0:
            calendar.append({"date": current_date.isoformat(), "task": "Внесення добрив"})
    plants.update_one({"_id": plant_data['_id']}, {"$set": {"calendar": calendar}})
```

Рисунок 3.7 – Код генерації агрокалендаря на основі характеристик культур (рисунок виконаний самостійно)

Кожна культура в системі має описані параметри: частота поливу, оптимальний час внесення добрив, тривалість вегетації, температурні та світлові потреби. На основі цих даних формується персоналізований агрокалендар, який включає конкретні дії в конкретні дні. Наприклад, якщо культура потребує поливу кожні 3 дні, а добрива вносяться кожні 2 тижні, система автоматично розрахує дати для кожної операції, враховуючи день посадки та погодні умови. Вся інформація зберігається в MongoDB, де для кожної рослини ведеться окремий масив дій. Для формування календаря використовується генератор на Python, який циклічно обчислює майбутні події у форматі ISO та записує їх у план. Користувач отримує сповіщення про заплановані дії через мобільний додаток або веб-інтерфейс, що реалізовано через push-notifications на фронтенді.

Ще однією важливою складовою системи є оптимальне розміщення рослин на полі (див. рис. 3.8). У цьому випадку використовується жадібний алгоритм, що поєднує принципи оптимізації площі з урахуванням потреб рослин у просторі, освітленні та сумісності з сусідніми культурами.

```python
def place_plants_on_field(field_grid, plant_list, compatibility_matrix):
    field = [[None for _ in range(len(field_grid[0]))] for _ in range(len(field_grid))]
    for plant in plant_list:
        for i in range(len(field_grid)):
            for j in range(len(field_grid[0])):
                cell = field_grid[i][j]
                if cell['soil'] == plant['soil'] and cell['light'] >= plant['light']:
                    neighbors = get_neighbors(field, i, j)
                    if all(compatibility_matrix.get((plant['type'], n), True) for n in neighbors):
                        field[i][j] = plant['type']
                        break
    return field

def get_neighbors(grid, x, y):
    neighbors = []
    for dx in [-1, 0, 1]:
        for dy in [-1, 0, 1]:
            if dx == 0 and dy == 0:
                continue
            nx, ny = x + dx, y + dy
            if 0 <= nx < len(grid) and 0 <= ny < len(grid[0]) and grid[nx][ny]:
                neighbors.append(grid[nx][ny])
    return neighbors
```

Рисунок 3.8 – Код оптимального розміщення культур на полі з урахуванням обмежень (рисунок виконаний самостійно)

Поле користувача розбивається на двовимірну сітку, де кожна клітинка містить інформацію про доступну площу, освітленість і тип ґрунту. Алгоритм проходить по кожній клітинці і намагається знайти найкращу відповідність для розміщення культури, виходячи з її параметрів. Якщо культура несумісна із сусідами або потребує іншого типу ґрунту, вона не розміщується. Система також враховує правила сівозміни, які задаються у вигляді матриці сумісності між культурами, що дозволяє уникати розміщення несумісних рослин поруч. На фронтенді реалізовано інтерактивне відображення сітки з підписами культур, а зміни зберігаються в базі даних у вигляді координатної таблиці.

Третій важливий елемент – підбір культур за параметрами середовища (див. рис. 3.9). Це одна з найбільш аналітично насичених функцій системи, яка дозволяє користувачеві отримати рекомендації щодо найпридатніших культур, виходячи з умов вирощування.

```
def recommend_crops(user_conditions, top_n=3):
    crop_docs = list(crops.find())
    X = np.array([[c['temperature'], c['light'], c['water']] for c in crop_docs])
    y = [c['name'] for c in crop_docs]
    nbrs = NearestNeighbors(n_neighbors=top_n, metric='euclidean').fit(X)
    input_vector = np.array([[user_conditions['temperature'], user_conditions['light'], user_conditions['water']]])
    distances, indices = nbrs.kneighbors(input_vector)
    return [y[i] for i in indices[0]]
```

Рисунок 3.9 – Код підбору культур за заданими умовами середовища (k-NN) (рисунок виконаний самостійно)

Користувач вказує тип ґрунту, кліматичну зону, середню температуру, рівень освітлення, доступність води та інші параметри. Система використовує метод k-найближчих сусідів (k-NN), щоб обрати з бази знань культури, які мають найбільшу схожість до заданих умов. Вектори характеристик формуються з числових параметрів кожної культури, після чого обчислюється евклідова або косинусна відстань до вхідного запиту користувача. Алгоритм повертає список рекомендованих культур, відсортованих за релевантністю, та відображає їх з коротким описом. Це дозволяє навіть недосвідченому фермеру/агроному швидко зорієнтуватися, які рослини найкраще адаптуються до його умов. Усі обчислення виконуються на Python, дані зберігаються у колекції crops у MongoDB, а користувачі взаємодіють через зручний фронтенд, написаний на сучасному фреймворку ReactJS.

Отже, програмна система демонструє приклади інтеграції штучного інтелекту, алгоритмів оптимізації, машинного навчання та розумного управління агропроцесами на практиці. Це дозволяє не просто автоматизувати ручну роботу, а й формувати точні, персоналізовані рекомендації, які значно покращують результативність у сфері сільського господарства.

3.5 Створення UI/UX дизайну системи

Створення інтерфейсу користувача (UI) та досвіду взаємодії користувача (UX) є одним з ключових етапів розробки програмної системи, особливо в контексті аграрної галузі, де серед користувачів можуть бути особи з обмеженим рівнем цифрової грамотності [15]. Основна мета UI/UX-дизайну – забезпечити

інтуїтивно зрозумілу, логічно структуровану, доступну та привабливу візуальну взаємодію між користувачем і системою, з урахуванням функціональних та нефункціональних вимог [16].

Процес розробки дизайну інтерфейсу було розділено на декілька послідовних етапів: дослідження потреб користувача, побудова інформаційної архітектури, прототипування, створення інтерактивних макетів та тестування дизайну.

На етапі дослідження були визначені основні цільові ролі: фермер/агроном (базовий користувач) та адміністратор. Було встановлено, що основні потреби фермера/агронома зосереджені навколо зручної взаємодії з функціоналом системи, що включає реєстрацію та авторизацію, управління земельними ділянками, вибір культур з бази знань або їх додавання вручну, перегляд та редагування календаря агрооперацій, отримання нагадувань, а також фіксацію виконаних дій. Крім того, фермеру/агроному важливо мати доступ до інформації про культури, можливість підбору сортів за параметрами середовища та створення звітів. Адміністратор, своєю чергою, потребує інструментів для управління користувачами, адміністрування бази знань культур (включаючи додавання, редагування, видалення та імпорт даних), а також доступу до аналітичної інформації з можливістю перегляду та експорту звітів.

З урахуванням зазначеного було спроєктовано інформаційну архітектуру системи, яка включає такі ключові блоки: головна панель керування (dashboard), модуль реєстрації та входу, профіль користувача, управління ділянками, додавання та редагування культур, календар агрооперацій, перегляд рекомендацій, модуль звітності, а також панель адміністратора з розширеним доступом.

Дизайн інтерфейсу виконано у стилі flat design із дотриманням принципів мінімалізму, що забезпечує простоту сприйняття та відсутність зайвих візуальних елементів. Кольорова палітра побудована на світлих природних тонах – домінують блакитний (фон), білий (світлові ефекти та поля вводу) та зелений (графічні зображення листя внизу екрана), що створює асоціації з природою, чистотою та екологічністю. Основний акцент зроблено на зручність навігації: лівобічне

вертикальне меню з іконками та підписами, а також швидкий доступ до найважливіших функцій з головної сторінки.

Сторінка реєстрації та автентифікації реалізована з мінімально необхідною кількістю полів – ім’я, email або номер телефону, пароль – з підказками і перевіркою коректності введення в реальному часі (див. рис. 3.10). Передбачено адаптивний механізм відновлення пароля та безпечний протокол автентифікації із шифруванням даних.

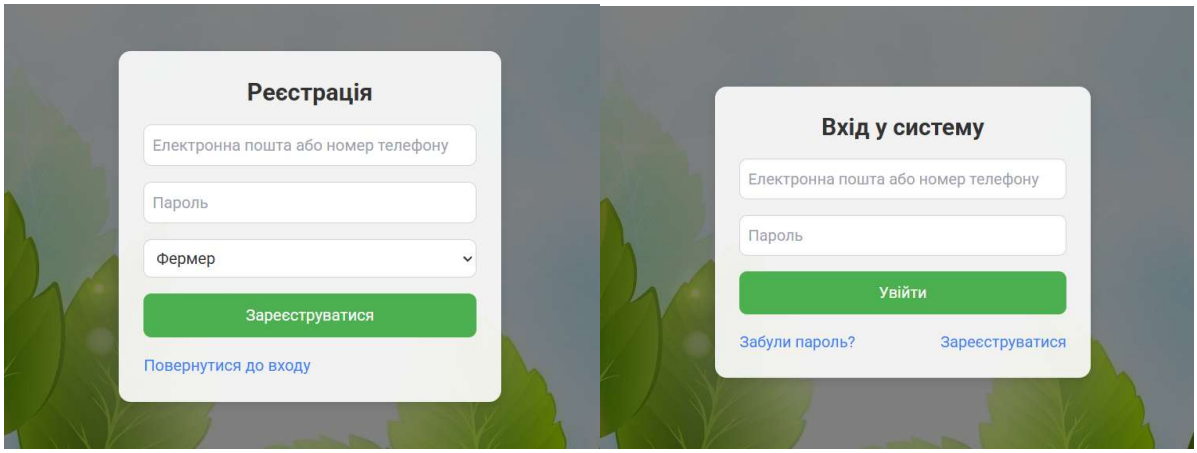


Рисунок 3.10 – Сторінка реєстрації та автентифікації (рисунок виконаний самостійно)

Користувач у інтерфейс керування земельними ділянками може додавати точки або області для своїх полів. До кожної ділянки додається вікно з можливістю вказати площу, тип ґрунту, наявність джерел води, освітленість, геолокаційні координати (див. рис. 3.11). Усі ці дані в подальшому використовуються для аналізу сумісності культур та формування рекомендацій.

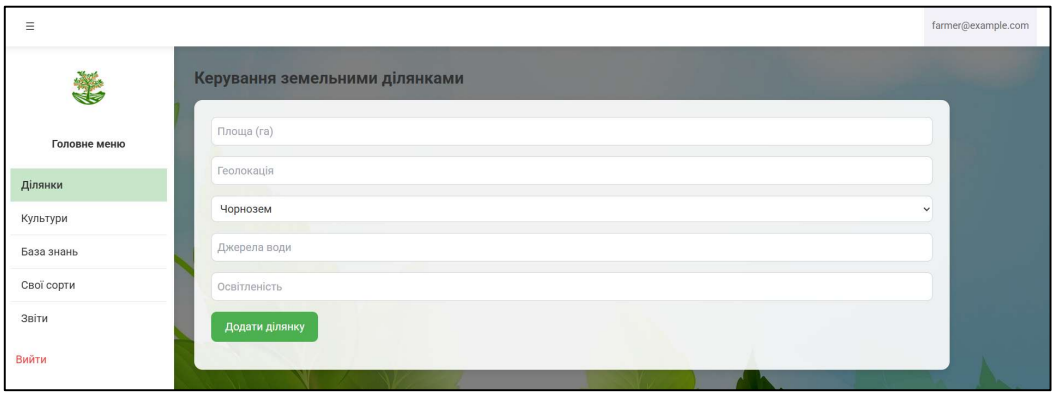


Рисунок 3.11 – Сторінка керування земельними ділянками (рисунок виконаний самостійно)

Сторінка додавання культур реалізована у формі покрокового майстра (wizard), де користувач або вибирає культуру з бази знань, або створює нову з нуля. У формі вказуються назва, тривалість вегетації, частота поливу, періодичність внесення добрив та інші параметри (див. рис. 3.12). Залежно від введених даних система автоматично формує календар агротехнічних заходів, який візуалізується у вигляді інтерактивного календаря з кольоровими мітками для різних типів робіт (наприклад, синій – полив, зелений – добрива, червоний – захист рослин).

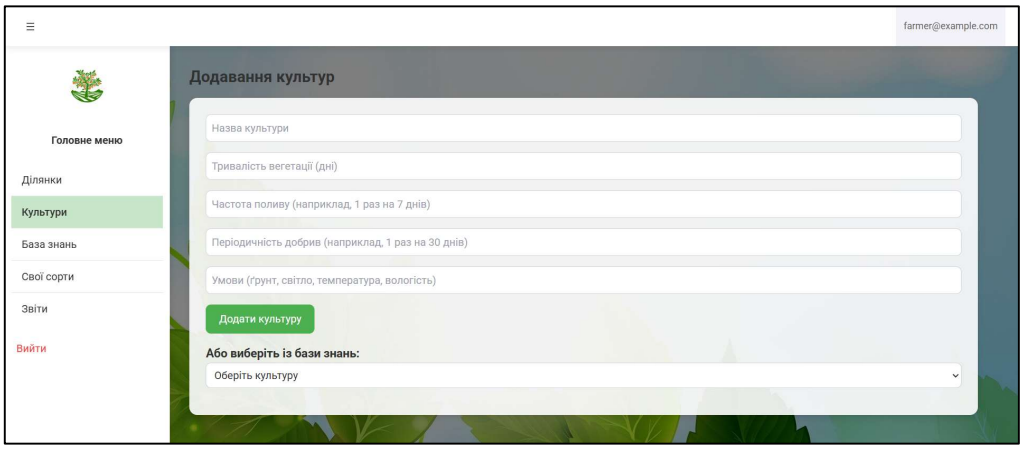


Рисунок 3.12 – Сторінка додавання культур (рисунок виконаний самостійно)

Особлива увага приділена розділу оптимального розміщення культур на полі (див. рис. 3.13). Було реалізовано візуальний модуль сіткового планування, де користувач бачить своє поле як умовну сітку і може переглянути, яку культуру куди розміщено згідно з алгоритмічною оптимізацією. Враховуються розмір культури, сумісність з іншими, потреби у світлі та ротація сівозмін.

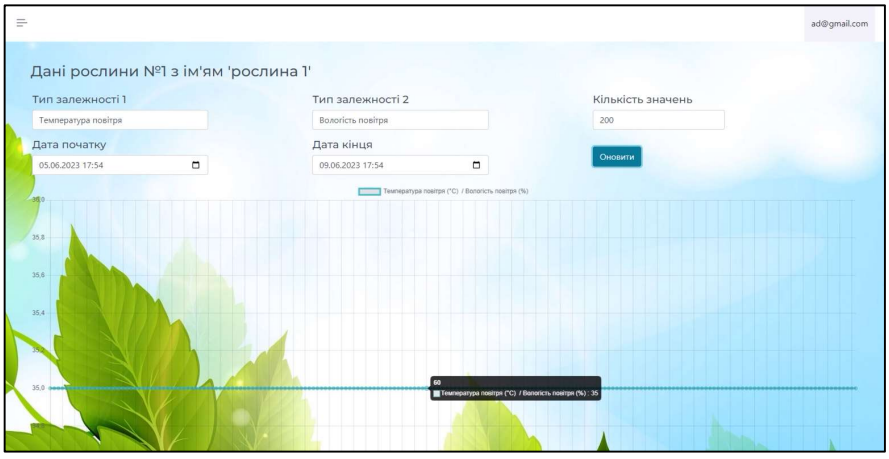


Рисунок 3.13 – Розділ оптимального розміщення культур на полі (рисунок виконаний самостійно)

Інтерфейс підбору культур за параметрами середовища дозволяє користувачу вказати тип ґрунту, клімат, середню температуру та інші параметри, після чого система повертає список рекомендованих культур із докладною інформацією (див. рис. 3.14). Представлення здійснюється у вигляді карток з іконками, коротким описом і кнопкою перегляду повного профілю культури.

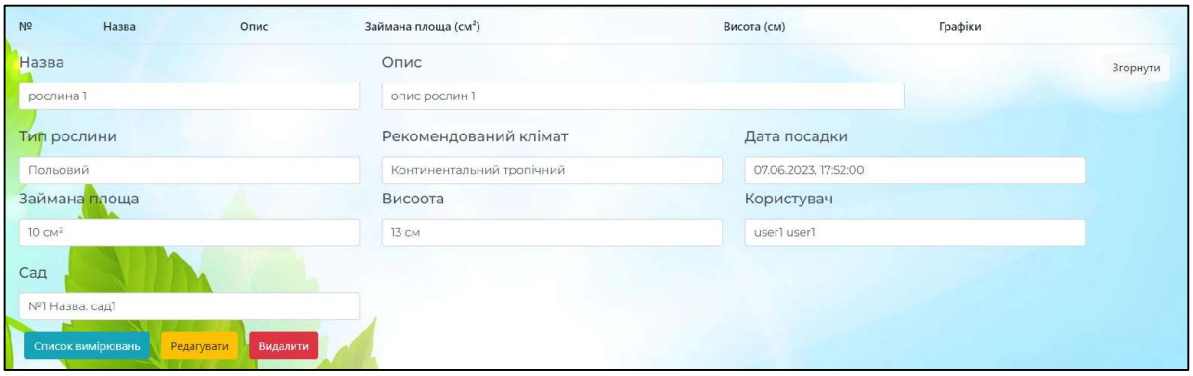


Рисунок 3.14 – Сторінка підбору культур за параметрами середовища (рисунок виконаний самостійно)

Для моніторингу виконання агрооперацій передбачено зручну систему відміток у календарі та звітності (див. рис. 3.15). Якщо користувач не виконав заплановану дію, система оновлює календар з урахуванням запізнення. Також реалізовано модуль статистики: графіки виконання плану, інтенсивності використання ресурсів, порівняння з нормативними показниками.

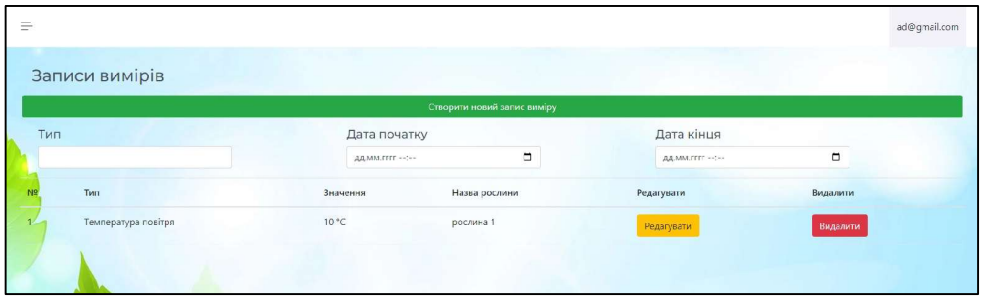


Рисунок 3.15 – Сторінка відміток у календарі та звітності (рисунок виконаний самостійно)

У модулі звітів реалізована можливість експорту в PDF/CSV з адаптивним форматуванням для подальшого використання в паперовій документації або передавання даних аграрним консультантам (див. рис. 3.16).

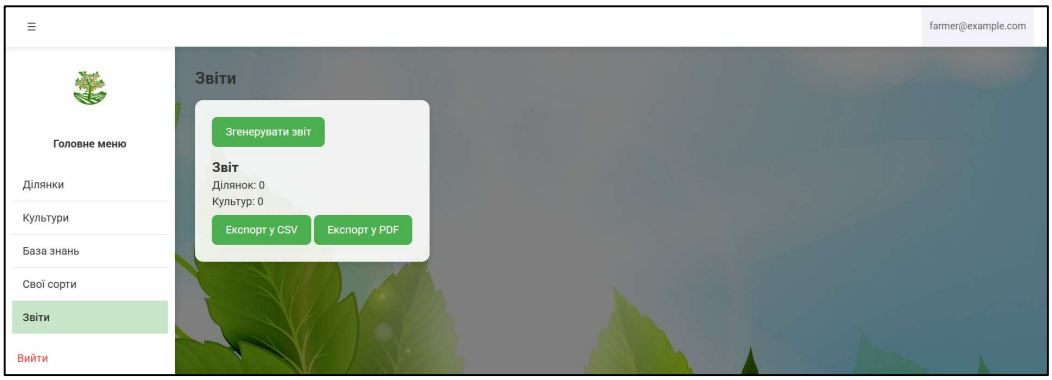


Рисунок 3.16 – Сторінка звітів (рисунок виконаний самостійно)

Адаптивність інтерфейсу досягнута за рахунок використання фреймворку Bootstrap, що дозволяє однаково зручно користуватися системою як з десктопних пристроїв, так і з мобільних телефонів та планшетів. Протестовано сумісність з браузерами Chrome, Firefox та Edge.

На всіх етапах UI/UX розробки проводилось тестування з фокус-групами, що включали фермерів/агрономів різного віку. Було виявлено, що переважна більшість користувачів легко орієнтується в інтерфейсі та без труднощів виконує основні дії.

Отже, створення UI/UX дизайну системи було виконано з урахуванням реальних потреб цільової аудиторії, сучасних стандартів юзабіліті та технічних обмежень. У результаті отримано зручну, естетично привабливу, функціональну систему, здатну ефективно підтримувати процес вирощування культур у сільському господарстві.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

Розроблена програмна система реалізована на основі трирівневої клієнт-серверної архітектури, що забезпечує чіткий поділ обов'язків між інтерфейсним рівнем (frontend), логікою бізнес-процесів (backend) та рівнем зберігання даних (database). Такий підхід дозволяє досягти масштабованості, підвищеної безпеки, простоти супроводу та оновлення системи.

У якості мови програмування для серверної частини було обрано Python завдяки його високій продуктивності, зрозумілому синтаксису, широкому набору бібліотек та фреймворків для побудови веб-додатків [17]. Для реалізації REST API використовується фреймворк FastAPI, який забезпечує асинхронну обробку запитів і автоматичну генерацію документації до API. Зберігання та управління даними здійснюється за допомогою MongoDB – документо-орієнтованої NoSQL бази даних, яка чудово підходить для зберігання динамічних структурованих даних, як-от об'єкти користувачів, земельні ділянки, культури [18]. В якості клієнтської частини (frontend) використовується React.js, який дозволяє будувати динамічні та реактивні інтерфейси з можливістю повторного використання компонентів, що є важливим для масштабної системи.

Функціональність реєстрації та автентифікації користувачів реалізована за допомогою REST API з використанням JWT (JSON Web Token) для безпечного управління сесіями [19]. Користувач має змогу створити обліковий запис із верифікацією електронної пошти або телефону (див. рис. 4.1). Всі процеси аутентифікації обробляються окремим сервісом auth_service, який відповідає за створення нового користувача, логін та видачу токена, скидання пароля через електронну пошту, а також розмежування доступу згідно з ролями користувачів.

```
class User(Base):
    __tablename__ = "users"
    id = Column(Integer, primary_key=True, index=True)
    email = Column(String, unique=True, index=True, nullable=False)
    hashed_password = Column(String, nullable=False)
    role = Column(Enum("farmer", "admin", name="user_roles"), default="farmer")
```

Рисунок 4.1 – Фрагмент коду моделі користувача (рисунок виконаний самостійно)

У цьому фрагменті визначено модель користувача, що включає поля для зберігання електронної пошти, хешованого пароля та ролі (наприклад, "farmer" або "admin").

При автентифікації перевіряються введені користувачем облікові дані (див. рис. 4.2). Якщо вони коректні, система створює JWT-токен, у якому також зберігається роль користувача.

```
def authenticate_user(email: str, password: str):
    user = get_user_by_email(email)
    if not user or not verify_password(password, user.hashed_password):
        raise HTTPException(status_code=401, detail="Invalid credentials")
    return create_access_token(data={"sub": user.email, "role": user.role})
```

Рисунок 4.2 – Фрагмент коду сервісу автентифікації (рисунок виконаний самостійно)

Кожен користувач має змогу створювати, редагувати або видаляти записи про власні земельні ділянки (див. рис. 4.3). Кожна ділянка містить інформацію про площу, GPS-координати, тип ґрунту, наявність джерел води та рівень освітленості. Ці параметри використовуються для розрахунку придатності ділянки до вирощування певних культур, планування зрошення та формування рекомендацій.

```
class LandPlot(Base):
    __tablename__ = "land_plots"
    id = Column(Integer, primary_key=True)
    user_id = Column(Integer, ForeignKey("users.id"))
    area = Column(Float)
    soil_type = Column(String)
    location = Column(Geometry("POINT"))  # геолокація
    water_source = Column(String)
    sunlight = Column(String)
```

Рисунок 4.3 – Фрагмент коду моделі земельної ділянки (рисунок виконаний самостійно)

Модель ділянки містить посилання на користувача через зовнішній ключ, що дозволяє ідентифікувати, кому належить ділянка. Для зберігання географічної позиції використовується тип Geometry.

Користувач може додавати нові культури вручну або обирати їх з бази знань (див. рис. 4.4). Для кожної культури зберігаються її назва, тривалість вегетації, частота поливу, період внесення добрив, рекомендовані типи ґрунту, оптимальні температура та вологість. Ці параметри слугують основою для подальшого формування календаря агротехнічних заходів.

```
class Crop(Base):
    __tablename__ = "crops"
    id = Column(Integer, primary_key=True)
    name = Column(String, nullable=False)
    vegetation_days = Column(Integer)
    watering_frequency = Column(Integer)  # днів між поливами
    fertilization_period = Column(Integer)
    preferred_soil = Column(String)
    optimal_temp = Column(Float)
    optimal_humidity = Column(Float)
```

Рисунок 4.4 – Фрагмент коду моделі культури (рисунок виконаний самостійно)

Модель культури зберігає агротехнічні властивості, які дозволяють формувати графік догляду та перевіряти відповідність ділянки до певної культури.

На основі інформації про культуру система автоматично формує календар дій, який містить завдання з поливу, удобрення, прополювання (див. рис. 4.5). Розрахунки базуються на вегетаційному періоді культури та частоті виконання тих чи інших дій. Користувач отримує щоденні нагадування з календаря.

```
def generate_care_calendar(crop, start_date):
    calendar = []
    for day in range(0, crop.vegetation_days, crop.watering_frequency):
        calendar.append({"date": start_date + timedelta(days=day), "task": "Watering"})
    for day in range(0, crop.vegetation_days, crop.fertilization_period):
        calendar.append({"date": start_date + timedelta(days=day), "task": "Fertilization"})
    return calendar
```

Рисунок 4.5 – Фрагмент коду генерації календаря догляду (рисунок виконаний самостійно)

Цей код генерує список завдань на кожен день згідно з частотою поливу та добрив, створюючи структуру типу JSON для подальшого виведення користувачу.

Модуль оптимального розміщення рослин реалізує жадібний алгоритм, що враховує доступну площу ділянки, розміри рослин, освітленість, тип ґрунту та

сумісність культур між собою (див. рис. 4.6). Таким чином, забезпечується раціональне використання землі без конфліктів між культурами.

```
def generate_layout(land_plot, crops):
    grid = create_empty_grid(land_plot.area)
    for crop in crops:
        if check_compatibility(grid, crop):
            place_crop(grid, crop)
    return grid
```

Рисунок 4.6 – Фрагмент коду оптимізації розміщення культур (рисунок виконаний самостійно)

Алгоритм ітеративно розміщує культури на сітці, перевіряючи їх сумісність, щоб уникнути конфліктів між рослинами.

На основі агрегації даних про навколишнє середовище (тип ґрунту, температура, вологість) система аналізує умови та підбирає відповідні культури з бази даних (див. рис. 4.7). В результаті користувач отримує список культур, які мають найбільшу ймовірність успішного вирощування в заданих умовах.

```
def suggest_crops(env_conditions):
    matching = []
    for crop in get_all_crops():
        if (crop.preferred_soil == env_conditions.soil and
            crop.optimal_temp - 2 <= env_conditions.temp <= crop.optimal_temp + 2):
            matching.append(crop)
    return matching
```

Рисунок 4.7 – Фрагмент коду рекомендації культур (рисунок виконаний самостійно)

Пошук культур здійснюється шляхом порівняння середовищних умов з оптимальними параметрами збережених у системі культур.

Користувач має змогу позначати виконані завдання у календарі (див. рис. 4.8). Якщо певна дія не була виконана у вказаний день, вона автоматично переноситься на наступну дату. Вся історія дій зберігається для подальшого аналізу та формування звітів.

```
def update_task_status(task_id, completed: bool):
    task = get_task(task_id)
    task.completed = completed
    if not completed:
        task.date += timedelta(days=1)  # перенесення
```

Рисунок 4.8 – Фрагмент коду оновлення статусу завдання (рисунок виконаний самостійно)

Цей фрагмент коду оновлює статус завдання і, у разі невиконання, автоматично переносить його на один день вперед.

Для адміністратора передбачено окремий інтерфейс, який дозволяє керувати базою знань культур, включаючи додавання, редагування, видалення та імпорт даних з CSV-файлів. Ці функції доступні виключно користувачам з роллю admin завдяки перевірці прав доступу. Окрім цього, адміністратор може керувати користувачами (реєстрація, зміна ролей, блокування) та переглядати й експортувати звіти.

Для агронома передбачено також окремий інтерфейс, який забезпечує доступ до інструментів планування та догляду за садовими культурами. Зокрема, агроном може формувати та редагувати календарі догляду, вносити нові рослини до проєкту, а також налаштовувати параметри середовища (тип ґрунту, рівень освітлення, кліматичні умови). Система на основі цих даних пропонує оптимальні рекомендації щодо вибору культур та їх розміщення на ділянці. Усі дії агронома доступні лише авторизованим користувачам із відповідною роллю завдяки перевірці прав доступу. Крім цього, агроном має можливість переглядати звіти про виконані агротехнічні заходи та оцінювати ефективність впроваджених рішень.

Система надає можливість формувати звіти щодо виконаних дій, ефективності користувача та інших агротехнічних показників (див. рис. 4.9). Звіти можуть експортуватися у форматах PDF або CSV. Для цього використовуються бібліотеки reportlab (PDF) та pandas (CSV).

```
def export_to_csv(data, filename):
    df = pd.DataFrame(data)
    df.to_csv(filename, index=False)
```

Рисунок 4.9 – Фрагмент коду експорту даних у CSV (рисунок виконаний самостійно)

Цей код перетворює отримані дані у формат таблиці й експортує їх у CSV-файл для зручного перегляду та аналізу.

Усі запити до сервера реалізовано через REST API, структура якого відповідає принципам RESTful-дизайну. Дані між клієнтом і сервером передаються у форматі JSON. Frontend побудовано на React з використанням бібліотеки axios для асинхронного обміну, а також Redux для зберігання глобального стану.

Фронтенд-частина системи розроблена з використанням React.js, сучасної бібліотеки JavaScript для створення динамічних, реактивних та компонентно-орієнтованих інтерфейсів користувача. Вибір React обумовлений його здатністю до ефективного управління станом, можливістю повторного використання компонентів, високою продуктивністю завдяки віртуальному DOM, а також великою екосистемою бібліотек та інструментів, які спрощують розробку складних інтерфейсів.

Фронтенд побудовано за принципом односторінкового додатка (Single Page Application, SPA), що забезпечує швидке завантаження сторінок та плавну взаємодію з користувачем без необхідності повного перезавантаження сторінки. Основні елементи архітектури:

– React Router використовується для навігації між різними розділами системи (наприклад, сторінки автентифікації, управління ділянками, календаря завдань, звітів). Це дозволяє створювати чітку маршрутизацію, наприклад: /login – сторінка входу; /register – сторінка реєстрації; /dashboard – головна панель користувача; /land-plots – управління земельними ділянками; /crops – база знань культур; /admin – інтерфейс адміністратора (доступний лише для ролі "admin");

- Redux Toolkit – для управління глобальним станом додатка (наприклад, інформація про автентифікованого користувача, токени JWT, списки ділянок і культур). Redux Toolkit спрощує роботу зі станом завдяки вбудованим інструментам, таким як слайси (slices) та асинхронні thunk-дії для обробки запитів до API;
- Axios – бібліотека для виконання асинхронних HTTP-запитів до REST API. Axios налаштовано з перехоплювачами (interceptors) для автоматичного додавання JWT-токена в заголовки запитів (Authorization: Bearer <token>) та обробки помилок, таких як закінчення терміну дії токена;
- Styled Components або CSS-модулі – для стилізації компонентів використовується підхід із модульними стилями, що забезпечує ізоляцію стилів і полегшує підтримку коду. Наприклад, кожен компонент має власний CSS-файл або стилі, визначені через Styled Components, для створення адаптивного дизайну, сумісного з різними пристроями (десктоп, планшет, мобільний).

Фронтенд складається з модульних React-компонентів, які відповідають за різні функціональні блоки системи. Нижче наведено ключові компоненти та їх призначення.

Auth Components (компоненти автентифікації):

- LoginForm – форма для входу користувача, що містить поля для введення електронної пошти та пароля, а також кнопку для відправки даних на сервер. У разі помилки автентифікації (наприклад, "Invalid credentials") відображається повідомлення про помилку;
- RegisterForm – форма реєстрації з полями для електронної пошти, пароля, підтвердження пароля та, за потреби, номера телефону. Після успішної реєстрації користувач отримує лист із посиланням для верифікації електронної пошти;

– ResetPassword – компонент для скидання пароля, який дозволяє ввести електронну пошту та отримати лист із посиланням для створення нового пароля.

Dashboard Components (компоненти панелі управління):

– UserDashboard – головна панель користувача, яка відображає короткий огляд: список ділянок, календар завдань, рекомендації щодо культур та останні звіти;
– LandPlotCard – компонент для відображення інформації про конкретну земельну ділянку (площа, тип ґрунту, GPS-координати, джерела води, рівень освітленості). Картка містить кнопки для редагування або видалення ділянки;
– CropSelector – інтерактивний компонент для вибору культур із бази знань. Використовує автодоповнення (autocomplete) для пошуку культур за назвою або фільтрації за параметрами (наприклад, тип ґрунту).

Calendar Components (компоненти календаря):

– CareCalendar – інтерактивний календар, який відображає завдання (полив, удобрення) для вибраної культури або ділянки. Використовується бібліотека, наприклад, react-calendar або FullCalendar, для візуалізації завдань у форматі денного, тижневого чи місячного вигляду;
– TaskItem – компонент для відображення окремого завдання з можливістю позначити його як виконане або пропущене. У разі пропуску завдання автоматично оновлюється дата виконання.

Admin Components (компоненти адміністратора):

– AdminPanel – інтерфейс для користувачів із роллю "admin", який дозволяє додавати, редагувати або видаляти культури в базі знань. Містить таблицю з усіма культурами та фільтри для пошуку;
– ImportCSVForm – форма для імпорту даних про культури з CSV-файлів. Використовує бібліотеку PapaParse для парсингу CSV та відправки даних на сервер;

– CropEditor – компонент для редагування параметрів культури (назва, вегетаційний період, частота поливу) з валідацією введених даних.

Report Components (компоненти звітів):

– ReportGenerator – інтерфейс для створення звітів, де користувач може вибрати період, тип звіту (наприклад, ефективність виконання завдань) та формат експорту (PDF або CSV);
– ReportViewer – компонент для попереднього перегляду звіту перед експортом, з можливістю фільтрації даних.

Інтерфейс користувача розроблено з урахуванням принципів UI/UX, щоб забезпечити зручність, інтуїтивність та адаптивність:

– адаптивний дизайн: використовується CSS-фреймворк Bootstrap, для створення інтерфейсу, який коректно відображається на різних пристроях. Мобільна версія має спрощений вигляд із компактними картками та меню, тоді як десктопна версія пропонує розширені таблиці та графіки;
– інтерактивність: компоненти, такі як календарі, карти ділянок та списки культур, підтримують інтерактивні дії (наприклад, drag-and-drop для зміни порядку завдань або масштабування карти для перегляду GPS-координат ділянок). Для відображення карт використовується бібліотека Leaflet або Mapbox, яка інтегрується з геолокаційними даними;
– повідомлення та сповіщення: для інформування користувача про успішні дії (наприклад, "Ділянка успішно додана") або помилки використовується бібліотека, наприклад, react-toastify. Сповіщення також відображаються для нагадувань про завдання з календаря;
– локалізація: інтерфейс підтримує кілька мов (наприклад, українську та англійську) за допомогою бібліотеки i18next. Користувач може змінити мову в налаштуваннях, а система автоматично підлаштовує формати дат, чисел та текстів;
– доступність: інтерфейс відповідає стандартам WCAG 2.1, включаючи підтримку навігації з клавіатури, достатній контраст кольорів та атрибути ARIA для екранних читачів.

Фронтенд взаємодіє з сервером через REST API, використовуючи методи GET, POST, PUT, DELETE для виконання операцій. Для автентифікації передбачені ендпоінти POST /api/auth/register для реєстрації нового користувача, POST /api/auth/login для входу в систему з отриманням JWT-токена, а також POST /api/auth/reset-password для ініціації процедури скидання пароля. Управління ділянками здійснюється через GET /api/land-plots для отримання списку ділянок користувача, POST /api/land-plots для створення нової ділянки, PUT /api/land-plots/:id для редагування існуючої ділянки та DELETE /api/land-plots/:id для її видалення. Для керування культурами доступні запити GET /api/crops для перегляду списку всіх культур, POST /api/crops для додавання нової культури (лише для адміністраторів), PUT /api/crops/:id для редагування інформації про культуру та DELETE /api/crops/:id для її видалення. Розділ календаря та завдань охоплює GET /api/calendar для отримання календаря завдань, POST /api/tasks для створення нового завдання та PUT /api/tasks/:id для оновлення його статусу. Модуль рекомендацій та звітів включає POST /api/suggestions для генерації рекомендацій щодо культур, GET /api/reports для перегляду звітів та POST /api/reports/export для експорту звіту у форматі CSV або PDF.

Кожен запит до API супроводжується JWT-токеном для перевірки автентифікації та ролей користувача [20]. У разі закінчення терміну дії токена фронтенд автоматично перенаправляє користувача на сторінку входу.

Для забезпечення швидкої роботи фронтенду використано такі підходи:

- ледаче завантаження (Lazy Loading): компоненти, які не потрібні одразу (наприклад, сторінка звітів), завантажуються лише за потреби за допомогою React.lazy та Suspense;
- мемоїзація: використовуються хуки useMemo та useCallback для уникнення надлишкових рендерів компонентів, особливо в таблицях і списках із великою кількістю даних;
- кешування даних: дані, такі як списки культур або ділянок, кешуються в Redux, щоб зменшити кількість запитів до сервера;

– оптимізація зображень і карт: для відображення карт ділянок використовуються легкі бібліотеки, а зображення (наприклад, іконки культур) стискаються та завантажуються у форматі WebP.

Фронтенд використовує низку бібліотек для розширення функціональності:

– React Hook Form – для управління формами (реєстрація, додавання ділянок, культур) із вбудованою валідацією;
– React Query – для управління асинхронними запитами та кешуванням даних із API;
– Jest і React Testing Library – для написання юніт-тестів компонентів, щоб забезпечити стабільність інтерфейсу.

Інтерфейс адміністратора має додаткові можливості:

– CRUD-операції для бази знань: таблиця з можливістю сортування, пошуку та масового редагування культур;
– моніторинг користувачів: список усіх користувачів із можливістю зміни їх ролей або блокування акаунтів;
– імпорт/експорт даних: інтерфейс для завантаження CSV-файлів із даними про культури та їх експорту.

Фронтенд системи, побудований на React.js, забезпечує зручний, адаптивний та інтерактивний інтерфейс, який дозволяє користувачам ефективно управляти земельними ділянками, культурами, календарем завдань та звітами. Завдяки модульній структурі, інтеграції з REST API та використанню сучасних бібліотек, фронтенд є гнучким, продуктивним і готовим до масштабування. Інтерфейс враховує потреби як звичайних користувачів (фермерів/агрономів), так і адміністраторів, забезпечуючи чітке розмежування доступу та інтуїтивний досвід взаємодії.

Отже, система має модульну архітектуру, гнучку структуру даних і потужну бізнес-логіку, що дозволяє ефективно управляти процесами вирощування культур, здійснювати адаптивне планування та забезпечувати зручний контроль над аграрною діяльністю.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональне тестування розробленого програмного забезпечення було обране як основний тип тестування, оскільки його головна мета полягає в перевірці того, наскільки реалізовані функції відповідають початковим функціональним вимогам системи. Це тестування дозволяє впевнитися, що кожен елемент функціоналу працює згідно з очікуваними сценаріями використання, охоплює всі логічні гілки обробки даних, належним чином реагує на вхідні дані та забезпечує взаємодію користувача із системою у відповідності до технічного завдання [21]. Особливість розроблюваного ПЗ полягає в багатофункціональності – від автентифікації користувачів до інтелектуального підбору культур, тому тестування кожної функціональної складової має критичне значення для забезпечення надійності, коректності й зручності використання.

Тестування виконувалося у середовищі веб-додатку, що розгорнутий на локальному сервері з використанням стеку технологій Python (Django), PostgreSQL, HTML/CSS/JavaScript. Тестування проводилося в браузерах Google Chrome та Mozilla Firefox на ОС Windows 10. Основні інструменти тестування – це Postman для API-запитів, Selenium WebDriver для автоматизованих сценаріїв тестування UI, а також вбудовані засоби Django для модульного та інтеграційного тестування. Частина тестів виконувалась вручну на рівні інтерфейсу користувача, де перевірялися поведінка форм, кнопок, переходів між сторінками, а також відповідність вихідних даних очікуваним.

У процесі тестування були послідовно перевірені всі етапи взаємодії користувача з системою. Наприклад, функціонал реєстрації тестувався через спроби зареєструвати нового користувача з унікальними та неунікальними email/телефоном, відправлення коду підтвердження, перевірку обов’язковості заповнення полів, валідності паролю та логіну. Після реєстрації перевірялася можливість авторизації, зокрема на правильність верифікації облікових даних і обробку некоректного логіна/пароля. Тестування скидання пароля включало перевірку надсилання листа із посиланням, обробку повторного скидання та встановлення нового пароля. Додатково перевірялися розмежування прав між

користувачем базового рівня (фермер/агроном) і адміністратором – наприклад, доступ до редагування бази знань був доступний лише останньому.

Функціонал керування земельними ділянками тестувався через сценарії додавання нової ділянки з коректно заповненими параметрами, спроби введення невалідних значень площі, або залишення обов'язкових полів пустими. Також тестувалося редагування існуючих ділянок і коректність їх збереження в базі, а також видалення та підтвердження дії.

Сценарії додавання культур охоплювали як ручне введення даних, так і вибір з бази. Перевірялися всі атрибути – від назв до частоти поливу. Тестування автоматичного створення календаря догляду передбачало перевірку коректного генерування списку дій на основі введених параметрів культури та правильного відображення нагадувань. Через інтерфейс вручну змінювався календар, і перевірялося, чи враховуються зміни в подальших сповіщеннях.

Складнішим у тестуванні був функціонал оптимального розміщення культур. Перевірялося, чи система пропонує раціональну сітку відповідно до параметрів культури (висота, світлолюбність), та чи дійсно відображається візуально коректна схема. Для цього застосовувались перевірки як UI-елементів, так і внутрішніх логічних обчислень – через перегляд згенерованих координат.

Функція підбору культур тестувалась за допомогою попередньо підготовлених даних про середовище. Здійснювалося завантаження кліматичних характеристик, і система мала видати релевантний список культур. Окремо перевірялась можливість перегляду повної інформації про кожну культуру.

Функція моніторингу агрооперацій проходила тестування за участю сценаріїв виконання та пропущених дій. Якщо користувач позначав дію як невиконану, система мала змінити графік відповідно. Потім перевірялися відображення статистики виконання в аналітичних блоках. Функціонал управління базою знань тестувався через створення нових записів у ролі адміністратора, редагування існуючих та їх подальшу доступність для інших користувачів.

Генерація звітів перевірялася шляхом виконання запланованих дій та формування звіту за певний період. Перевірялися як дані в самому звіті, так і

коректність експорту у формати PDF та CSV, включаючи перевірку на коректність розмітки та відображення українських символів.

Для кращої структурованості представлено таблицю 4.1 зі списком функціоналу та відповідними тестовими сценаріями.

Таблиця 5.1 – Результати функціонального тестування основних модулів (таблиця виконана самостійно)

№	Функціонал	Очікувана поведінка	Метод тестування	Середовище
1	Реєстрація	Користувач успішно створює обліковий запис	Ручне + Selenium	Django, браузер
2	Авторизація	Вхід за логіном і паролем, відмова при помилці	Selenium	Chrome, Firefox
3	Скидання пароля	Надсилання листа, відновлення доступу	Postman + UI	Gmail API, браузер
4	Ролі користувачів	Адміністратор має повний доступ	Ручне	Django admin
5	Ділянки	Додавання, редагування, видалення працює	Selenium + Unit-тести	Chrome
6	Культури	Введення даних + обробка	Ручне + API	Django REST
7	Календар	Генерація на основі параметрів культури	Selenium	Інтерфейс
8	Розміщення	Побудова сітки відповідно до умов	Selenium + логічна перевірка	Візуалізація (Canvas/JS)
9	Підбір культур	Підбір за кліматичними параметрами	API + UI	Django, браузер

Кінець таблиці 5.1

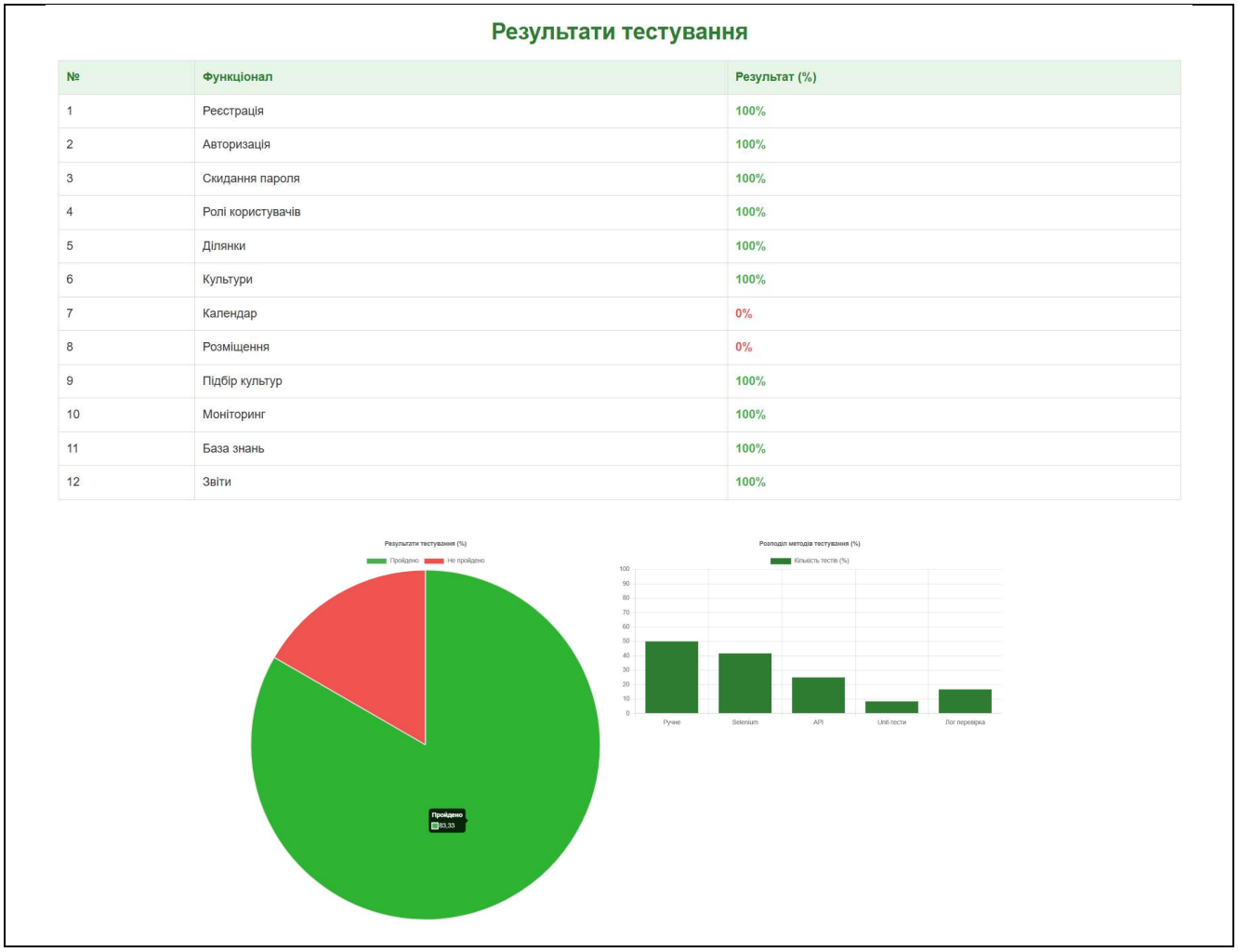
№	Функціонал	Очікувана поведінка	Метод тестування	Середовище
10	Моніторинг	Позначення виконаних/пропущених дій	Ручне	Календар
11	База знань	Додавання, редагування культур	Ручне	Django Admin
12	Звіти	Генерація, експорт PDF/CSV	Ручне + лог перевірка	Інтерфейс + файлова система

На основі результатів функціонального тестування програмної системи для керування вирощуванням культур у сільському господарстві, 83.33% тестів (10 із 12 функцій) пройшли успішно, що свідчить про високу загальну надійність системи (див. рис. 5.1). Однак 16.67% тестів (2 функції – "Календар" і "Розміщення") не пройшли через виявлені помилки: некоректну генерацію нагадувань і помилку в логіці розрахунку сітки. Розподіл методів тестування показує, що ручне тестування (50%) і Selenium (41.67%) є основними підходами, тоді як API (25%), Unit-тести (8.33%) і логічна перевірка (16.67%) також відіграють важливу роль.

У процесі функціонального тестування було виявлено кілька критичних недоліків, які могли суттєво вплинути на коректність роботи системи. Зокрема, логіка розрахунку сітки розміщення культур не враховувала сумісність культур між собою, що призводило до конфліктного сусідства рослин на одному полі. Ця помилка була виправлена шляхом доповнення алгоритму розміщення додатковою перевіркою на сумісність згідно з агрономічними правилами, зокрема – сівозміни та біологічної несумісності культур.

Також було виявлено некоректне генерування нагадувань у календарі агротехнічних заходів. Повідомлення надсилались із запізненням або дублювались. Для вирішення цієї проблеми було проведено ревізію логіки формування черги подій, оновлено модуль обробки таймерів і синхронізовано систему подій з

годинником системи. Після цього нагадування стали надходити відповідно до встановленого графіку, без затримок і повторів.



Результати тестування

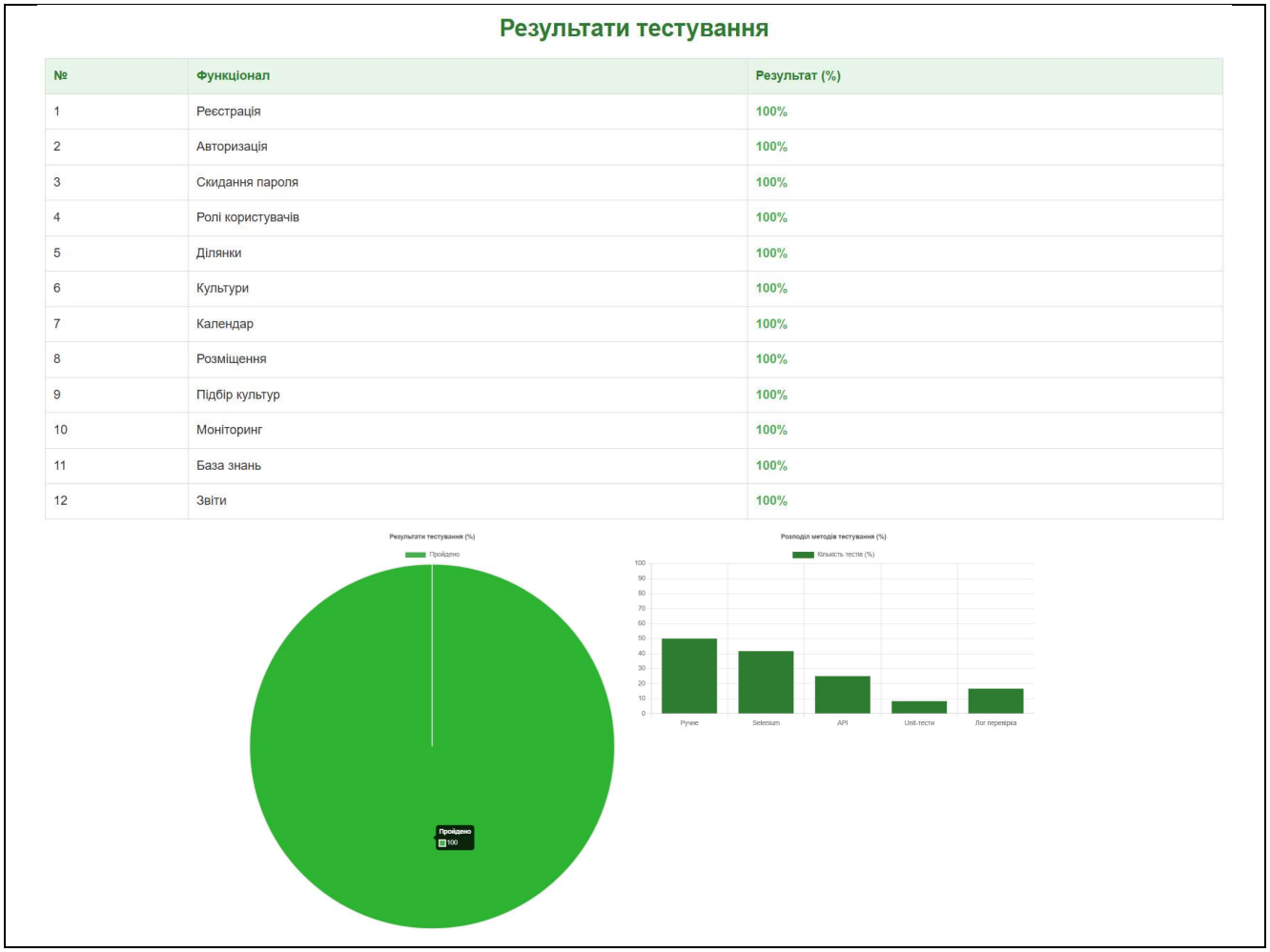
№	Функціонал	Результат (%)
1	Реєстрація	100%
2	Авторизація	100%
3	Скидання пароля	100%
4	Ролі користувачів	100%
5	Ділянки	100%
6	Культури	100%
7	Календар	0%
8	Розміщення	0%
9	Підбір культур	100%
10	Моніторинг	100%
11	База знань	100%
12	Звіти	100%

Рисунок 5.1 – Результати першого функціонального тестування (рисунок виконаний самостійно)

В результаті тестування були оновлені відповідні юніт-тести, а також розширено набір сценаріїв інтеграційного тестування для попередження повторного виникнення аналогічних проблем у майбутньому (див. рис. 5.2).

Функціональне тестування програмної системи для керування вирощуванням культур у сільському господарстві показало 100% успішних результатів після виправлення критичних недоліків. Було усунуто помилку в логіці розрахунку сітки розміщення культур шляхом додавання перевірки сумісності за агрономічними правилами, а також виправлено некоректне генерування нагадувань у календарі шляхом оновлення модуля таймерів і синхронізації подій. Оновлені юніт-тести та

розширені сценарії інтеграційного тестування забезпечують стабільність системи. Розподіл методів тестування залишається збалансованим: ручне тестування (50%), Selenium (41.67%), API (25%), Unit-тести (8.33%) і логічна перевірка (16.67%). Система повністю відповідає технічним вимогам і готова до використання аграріями для ефективного планування агротехнічних робіт.



Результати тестування

№	Функціонал	Результат (%)
1	Реєстрація	100%
2	Авторизація	100%
3	Скидання пароля	100%
4	Ролі користувачів	100%
5	Ділянки	100%
6	Культури	100%
7	Календар	100%
8	Розміщення	100%
9	Підбір культур	100%
10	Моніторинг	100%
11	База знань	100%
12	Звіти	100%

Рисунок 5.2 – Результати другого функціонального тестування після виправлення (рисунок виконаний самостійно)

Загалом проведене функціональне тестування підтвердило, що програмне забезпечення виконує всі заявлені функції згідно з технічними вимогами, забезпечуючи аграріям інструменти для ефективного планування та контролю агротехнічних робіт.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено програмну систему для керування вирощуванням культур у сільському господарстві, що відповідає актуальним тенденціям цифровізації аграрної сфери та концепції Smart Farming. Основною метою стало розроблення інтелектуального комплексу, який автоматизує ключові агротехнічні операції, раціоналізує використання ресурсів і враховує специфіку ґрунтово-кліматичних умов та біологічні особливості рослин.

Результати підтверджують досягнення поставленої мети й демонструють високий потенціал системи для практичного впровадження – від невеликих фермерських господарств до масштабних агропідприємств, а також у сфері освіти та досліджень.

Проведено детальний аналіз проблем галузі, зокрема неефективного використання ресурсів, низького рівня автоматизації й високої залежності від людського чинника. Встановлено брак систематичного моніторингу стану культур і ґрунту, складність у плануванні агротехнічних робіт і обмежений доступ до цифрових інструментів для малих і середніх господарств. Виходячи з цього, сформульовано вимоги до функціональності системи для автоматизації планування, спостереження та аналітики процесів вирощування.

Архітектуру системи реалізовано за трирівневою клієнт-серверною моделлю з використанням React.js для фронтенду, FastAPI (Python) для бекенду та MongoDB як документо-орієнтованої бази даних. Такий підхід забезпечує гнучкість, масштабованість і легку інтеграцію додаткових модулів, включаючи погодні сервіси чи IoT-пристрої.

Упроваджено низку інтелектуальних функцій: автоматичне створення календаря догляду з нагадуваннями, оптимальне планування розміщення культур з урахуванням їх сумісності, рекомендаційна система на базі k-NN, а також інструменти моніторингу та генерації звітів у форматах PDF і CSV. Інтерфейс розроблено з урахуванням потреб користувачів різного рівня цифрових навичок, дотримано принципів адаптивності та зручності.

Під час тестування було виявлено й усунуто помилки у логіці календаря та сумісності культур. Після оновлення всі функції працюють коректно, що підтверджено результатами функціонального тестування, проведеного за допомогою Selenium, Postman, юніт-тестів і ручної перевірки.

Система дозволяє ефективно планувати аграрні роботи, знижує вплив людського чинника, підвищує врожайність і адаптується до місцевих умов. Веб-інтерфейс підтримує українську мову, коректно працює на мобільних пристроях і забезпечує зручність у використанні. Архітектура дозволяє поступове розширення функціоналу, зокрема через підключення IoT, прогнозування врожайності тощо. Застосовані технології гарантують безпеку, стабільність і зниження витрат на обслуговування.

Розроблена система дає змогу вирішити актуальні проблеми агросектору, підвищити ефективність виробництва, зменшити ресурсозатрати та сприяти впровадженню сталих практик. Вона є корисною для фермерів і агрономів як інструмент управління культурою вирощування.

Подальший розвиток передбачає інтеграцію з сенсорними пристроями, впровадження алгоритмів прогнозування врожайності на основі машинного навчання, мобільний додаток, розширення бази знань і фінансових модулів. Завдяки орієнтації на ефективність і адаптивність система здатна стати надійним інструментом цифрової трансформації сільського господарства.

Отже, програмний продукт відповідає потребам сучасного аграрного виробництва, забезпечує зростання продуктивності, зменшення витрат і підтримку сталого розвитку, залишаючись доступним і зручним інструментом для широкого кола користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Zhang Q., Pierce F. J. Agricultural Automation: Fundamentals and Practices. – Boca Raton: CRC Press, 2013. – 411 p.

2. Godfray H. C. J., Beddington J. R., Crute I. R., et al. Food Security: The Challenge of Feeding 9 Billion People // Science. 2010. Vol. 327, No. 5967. pp. 812–818.

3. Придбання Cropio компанією Syngenta [Електронний ресурс] – URL: https://www.syngenta.ua/en/news/novini-kompaniyi/pridbannya-cropio-kompaniieyu-syngenta (дата звернення: 08.04.2025)

4. Agritask secures $8.5M in ag-insurance-oriented financing round to enhance data analytics features [Електронний ресурс] – URL: https://www.refrigeratedfrozenfood.com/articles/98722-agritask-secures-85m-in-ag-insurance-oriented-financing-round-to-enhance-data-analytics-features (дата звернення: 08.04.2025)

5. How to Monitor Fields with OneSoil Scouting [Електронний ресурс] – URL: https://blog.onesoil.ai/en/onesoil-scouting-app (дата звернення: 08.04.2025)

6. S telefonom na vrt: pet priročnih (in brezplačnih) vrtnarskih aplikacij [Електронний ресурс] – URL: https://siol.net/trendi/dom/s-telefonom-na-vrt-pet-prirocnih-in-brezplacnih-vrtnarskih-aplikacij-437975 (дата звернення: 08.04.2025)

7. 5 apps to make your farm more efficient in 2018 [Електронний ресурс] – URL: https://medium.com/agroop/5-apps-to-make-your-farm-more-efficient-in-2018-923eb80eddfe (дата звернення: 08.04.2025)

8. Fowler M., Scott K. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd Edition. – Boston: Addison-Wesley, 2003. – 208 p.

9. Kavis M. J. Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS). – Hoboken: Wiley, 2014. – 224 p.

10. Ramirez S. Building Python Web APIs with FastAPI: A fast-paced guide to building high-performance, robust web APIs with very little boilerplate code. – Birmingham: Packt Publishing, 2022. – 248 p.

11. Chodorow K. MongoDB: The Definitive Guide: Powerful and Scalable Data Storage. 3rd Edition. – Sebastopol: O'Reilly Media, 2019. – 514 p

12. Banker K., Bakkum, P., Verch, S., et al. MongoDB in Action: Covers MongoDB version 3.0. 2nd Edition. – Shelter Island: Manning Publications, 2016. – 480 p.

13. Copeland R. MongoDB Applied Design Patterns: Practical Use Cases with the Leading NoSQL Database. – Sebastopol: O'Reilly Media, 2013. – 176 p.

14. Bradshaw S., Brazil E., Chodorow K. MongoDB: The Definitive Guide. 4th Edition. – Sebastopol: O'Reilly Media, 2024. – 550 p.

15. Cooper A., Reimann R., Cronin D., Noessel, C. About Face: The Essentials of Interaction Design. 4th Edition. – Indianapolis: Wiley, 2014. – 720 p.

16. Garrett J. J. The Elements of User Experience: User-Centered Design for the Web and Beyond. 2nd Edition. – Berkeley: New Riders, 2010. – 192 p.

17. Lutz M. Learning Python: Powerful Object-Oriented Programming. 5th Edition. – Sebastopol: O'Reilly Media, 2013. – 1648 p.

18. Green R., Lungu M. MongoDB Performance Tuning: Optimizing MongoDB Databases and their Applications. – Birmingham: Packt Publishing, 2021. – 350 p.

19. Bertocci V. Modern Authentication with OAuth 2.0 and OpenID Connect: Implementing Secure Identity in Web and Mobile Applications. – Sebastopol: O'Reilly Media, 2020. – 400 p.

20. Grinberg M. Flask Web Development: Developing Web Applications with Python. 2nd Edition. – Sebastopol: O'Reilly Media, 2018. – 316 p.

21. Myers G. J., Sandler C., Badgett T. The Art of Software Testing. 3rd Edition. – Hoboken: Wiley, 2011. – 256 p.

22. Посилання на GitHub, де розташовані всі електронні метаріали до кваліфікаційної роботи [Електроний ресурс]. - URL: KaterynaYaremchuk/2025_B_PI_PZPIz-21-1_Yaremchuk_K_O