Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Лєпшакова Максима Руслановича*
(ПІБ)

академічної групи *122-19-4*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка комп'ютерної системи моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*проф. Лактіонов І.С.*			
розділів:				
спеціальний	*проф. Лактіонов І.С.*			
економічний	*проф. Вагонова О.Г.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу
бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-19-4* *Лєпшакова Максима Руслановича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка комп'ютерної системи моніторингу* сільськогосподарських виробництв рослинництва відкритого ґрунту

затверджена наказом ректора НТУ «ДП» від *16.05.2023* № *350-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів практик та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***19.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2023 р.***

Завдання видав *проф. Лактіонов І.С.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання *Лєпшаков М.Р.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *19.06*

РЕФЕРАТ

Пояснювальна записка: 71 с., 11 рис., 2 табл., 1 дод., 20 джерел.

Об'єкт дослідження – веб додаток для комп'ютера, призначенням якого є моніторинг даних сільськогосподарських полів.

Мета кваліфікаційної роботи – розробка комп’ютерної системи моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту задля підвищення ефективності та оптимізації процесів сільського господарства.

У вступі дослідження проводиться аналіз та опис сучасного стану проблеми, а також уточнена мета кваліфікаційної роботи та галузь її застосування. Крім того, надається обґрунтування актуальності обраної теми і більш детально сформульована постановка завдання.

У першому розділі проведено детальний аналіз предметної галузі, визначено актуальність поставленого завдання та цілей розробки, а також сформульовано постановку завдання з урахуванням вимог до програмної реалізації, необхідних технологій та програмних засобів.

У другому розділі проведений аналіз існуючих рішень, вибрані платформи для розробки, здійснено проектування та розробку програми. Описано роботу програми, включаючи алгоритм та структуру її функціонування. Також описано процес виклику та завантаження програми, визначено вхідні і вихідні дані, а також наведено опис параметрів технічних засобів.

У розділі, присвяченому економіці, було визначено трудомісткість розробки інформаційної системи, був оцінений час, необхідний для його реалізації та проведено розрахунок вартості роботи зі створення програми.

Практичне значення даної роботи полягає у розробці додатка, який надає можливість електронного зберігання, ведення бази даних та обліку полів, сільськогосподарських культур та інших важливих аспектів, що пов'язані з сільськогосподарським виробництвом.

Актуальність інформаційної системи визначається великим попитом на подібні розробки, які спрощують та оптимізують процес ведення бази даних в аграрному напрямку.

Список ключових слів: ІНФОРМАЦІЙНА СИСТЕМА, КОМП’ЮТЕР, ОБЛІК, БАЗА ДАНИХ, АВТОМАТИЗАЦІЯ, ПРОЕКТУВАННЯ, ДОДАТОК, МЕНЮ

ABSTRACT

Explanatory note: 71 pages, 11 pictures, 2 tables, 1 appendix, 20 sources.

The research object is a web application for a computer, the purpose of which is to monitor field data.

The purpose of the qualification work is to develop a computer system for monitoring agricultural productions of open ground crop production with the aim of increasing efficiency and optimizing agricultural processes.

In the introduction to the study, an analysis and description of the current state of the problem is carried out, as well as the purpose of the qualification work and the field of its application are clarified. In addition, a justification of the relevance of the chosen topic and a more detailed statement of the task are provided.

In the first chapter, a detailed analysis of the subject area was carried out, the relevance of the task and development goals was determined, and the task statement was formulated taking into account the requirements for software implementation, the necessary technologies and software tools.

In the second section, an analysis of existing solutions was carried out, platforms were selected for development, and the program was designed and developed. The operation of the program is described, including the algorithm and structure of its functioning. The process of calling and downloading the program is also described, input and output data are defined, and a description of technical means parameters is also given.

In the section devoted to the economy, the labor intensity of the development of the information system was determined, the time required for its implementation was estimated, and the cost of work on creating the program was calculated.

The practical significance of this work lies in the development of an application that provides the possibility of electronic storage, database management and accounting of fields, crops and other important aspects related to agricultural production.

The relevance of the information system is determined by the great demand for similar developments that simplify and optimize the process of maintaining a database in the agricultural sector.

Keywords: INFORMATION SYSTEM, COMPUTER, ACCOUNTING, DATABASE, AUTOMATION, DESIGN, APPENDIX, MENU.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

ОС – операційна система;

SDK – software development kit;

IDE – integrated development environment;

GUI – graphical user interface.

ЗМІСТ

ВСТУП

Швидкий розвиток технологій приніс численні досягнення в різних галузях, включаючи сільське господарство. У сільськогосподарському секторі потреба в ефективних системах моніторингу для оптимізації та впорядкування процесів виробництва сільськогосподарських культур стає все більш очевидною. У зв’язку з цим, Моя кваліфікаційна робота спрямована на створення веб додатку, що сприятиме спрощенню та оптимізації роботи у агросфері.

Об'єкт діяльності фахівця в галузі пов'язаний безпосередньо з сільським господарством, а спеціальність – комп’ютерні науки. Зв'язок між проблемою, яка вирішується, та об'єктом та спеціальністю полягає в необхідності створення системи, що дозволить ефективно моніторити і управляти сільськогосподарськими виробництвами відкритого ґрунту. Враховуючи набутий досвід виникає необхідність в розробці веб-системи, що враховує основні аспекти вирощування культур, управління полями, добривами та аналізом врожайності. Ця система має за мету відповідати конкретним потребам сільськогосподарської галузі, забезпечуючи фермерам і фахівцям зручне рішення для оптимізації процесів вирощування культур на відкритому ґрунті.

Мета цієї кваліфікаційної роботи полягає в розробці веб-системи для моніторингу сільськогосподарських виробництв відкритого ґрунту з використанням Express.js, React та MySQL. Основне призначення розробки – забезпечити фермерам та фахівцям інструмент для ефективного контролю та управління виробництвом на відкритому ґрунті.

Актуальність теми полягає в необхідності удосконалення робочих процедур в аграрній галузі, забезпечення ефективного зберігання та обробки інформації, підвищення якості моніторингу цих сфер. Зараз багато фермерів все ще покладаються на традиційні паперові або фрагментовані електронні системи, що призводить до неоптимального використання ресурсів, неточностей у веденні записів і незручностей для користувачів.

Ця кваліфікаційна робота передбачає розробку веб-додатку, який

забезпечує єдину систему управління даними про поля та сільськогосподарські культури. Функціонал додатку включає реєстрацію користувачів, ведення інформації про поля, добрива, сільськогосподарські культури та інші процеси. Для розробки зручного та інтуїтивно зрозумілого інтерфейсу користувача використовується мова програмування JavaScript та фреймворк REACT.

В якості бази даних для зберігання інформації про моніторинг полів була обрана реляційна БД – MySQL. MySQL — це широко використовувана та дуже популярна система керування реляційними базами даних із відкритим кодом. Він має міцну репутацію надійності, стабільності та продуктивності, що робить його придатним вибором для обробки даних інтенсивного характеру моніторингу сільськогосподарського виробництва.

MySQL пропонує чудову масштабованість, що дозволяє системі обробляти великі обсяги даних без втрати продуктивності. Це надзвичайно важливо для системи сільськогосподарського моніторингу, оскільки вона повинна обробляти величезну кількість інформації, пов’язаної з рослинництвом, включаючи врожайність, дані про ріст, погодні умови тощо..

Однією з ключових переваг MySQL є її здатність ефективно підтримувати складні запити. Система може обробляти розширені запити та агрегації, дозволяючи проводити поглиблений аналіз і звітувати про сільськогосподарські дані. Це важливо для отримання інформації та прийняття обґрунтованих рішень на основі зібраних даних.

Ще однією причиною вибору MySQL є його сумісність з іншими технологіями стеку веб-системи, включаючи Express.js і React. MySQL бездоганно інтегрується з цими технологіями, забезпечуючи надійну основу для зберігання та отримання даних. Ця сумісність забезпечує безперебійний потік даних і взаємодію між зовнішнім і внутрішнім компонентами веб-системи.

Таким чином, MySQL було обрано за її надійність, масштабованість, продуктивність запитів, сумісність з іншими технологіями, надійні функції безпеки та широку підтримку спільноти. Ці фактори роблять MySQL придатним і вигідним вибором для розробки веб-системи моніторингу

сільськогосподарського виробництва відкритого ґрунту.

Ця кваліфікаційна робота має на меті створення веб-додатку, який дозволить ефективно працювати з даними полів і поліпшити керування інформацією. Впровадження цього додатку в галузі фермерства сприятиме оптимізації робочих процесів і поліпшенню загальної продуктивності.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Комп'ютерні системи моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту є важливим інструментом для підтримки ефективного агрономічного управління. Вони дозволяють збирати, аналізувати і використовувати різноманітну інформацію про стан рослинництва, навколишнє середовище та інші фактори, що впливають на вирощування сільськогосподарських культур.

Сучасний стан проблеми використання комп'ютерних систем моніторингу відображається в ряді аспектів. Перш за все, було проведено аналіз аналогів, який показав, що існують різноманітні розробки і вироби, спрямовані на вирішення завдань моніторингу в рослинництві. Однак, не всі ці розробки досягли достатнього рівня розв'язання поставлених завдань. Існують технічні протиріччя, пов'язані з точністю збору інформації, швидкістю обробки даних та доступністю високоякісних сенсорів для вимірювання параметрів рослинництва.

Окрім того, прогалини знань в цій галузі можуть вплинути на ефективність використання комп'ютерних систем моніторингу. Важливо розвивати наукові та практичні дослідження, спрямовані на розуміння фізіологічних та екологічних аспектів рослинництва і використання цих знань для поліпшення систем моніторингу.

Нездійснені вимоги до виробів чи розробок наукового, організаційного або іншого характеру також впливають на розвиток комп'ютерних систем моніторингу. Наприклад, деякі вимоги можуть бути пов'язані з покращенням алгоритмів обробки даних, забезпеченням високошвидкісного збереження даних або встановленням стандартів для обміну інформацією між різними системами.

У галузі застосування програм або програмного виробу комп'ютерної

системи моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту використовуються для збору даних про рослини, такі як врожайність, фізіологічний стан, вологість, поживні речовини та інші показники. Ці системи допомагають агрономам та фермерам приймати інформовані рішення щодо поливу, внесення добрив, контролю за шкідниками та хворобами, а також визначення оптимального часу збору врожаю.

Об'єктом використання програм або програмного виробу є сільськогосподарські поля відкритого ґрунту, де здійснюється вирощування рослинництва. Вони можуть бути використані як на невеликих сільськогосподарських господарствах, так і на великих агропромислових підприємствах.

Окрім того, прогалини знань в цій галузі можуть вплинути на ефективність використання комп'ютерних систем моніторингу. Важливо розвивати наукові та практичні дослідження, спрямовані на розуміння фізіологічних та екологічних аспектів рослинництва і використання цих знань для поліпшення систем моніторингу.

Також важливо враховувати організаційні аспекти впровадження комп'ютерних систем моніторингу. Наприклад, необхідно забезпечити навчання та підтримку персоналу, який буде використовувати ці системи. Доступність і простота використання інтерфейсу також грають важливу роль у прийнятті рішень на основі зібраних даних.

Застосування комп'ютерних систем моніторингу в сільському господарстві має значний потенціал для підвищення ефективності та стійкості виробництва рослинництва відкритого ґрунту. Вони дозволяють здійснювати ретельний аналіз, передбачати ризики та вчасно реагувати на зміни в умовах вирощування. Це сприяє збільшенню врожайності, зниженню витрат на виробництво та збереженню ресурсів.

Проте, для подальшого розвитку комп'ютерних систем моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту необхідно звернути увагу на деякі виклики. Наприклад, інтеграція різних типів сенсорів і

засобів збору даних може стати складною задачею через відсутність єдиних стандартів. Також потрібно розвивати алгоритми обробки даних, щоб забезпечити точність та швидкість аналізу.

1.2. Призначення розробки та галузь застосування

Мета розробки цієї програми забезпечити комплексну систему сільськогосподарського моніторингу та управління для фермерів та агрономів. Основне призначення розробки полягає у тому, щоб користувачі могли приймати рішення на основі даних і оптимізувати сільськогосподарські методи для вирощування сільськогосподарських культур у відкритому грунті.

Сфера застосування для проекту буде в сільському господарстві, особливо зосереджуючись на моніторингу та управлінні виробництвом сільськогосподарських культур на відкритих полях. Він обслуговуватиму широкий спектр культур, вирощених у різних географічних регіонах. Система допоможе фермерам і агрономам у моніторингу ключових параметрів, таких як параметри поля, типи та параметри сільськогосподарських культур, добрива.

Метою програми буде забезпечення аналізу, візуалізації та звітності даних у режимі реального часу та історичних даних. Це дозволить користувачам відстежувати та аналізувати продуктивність своїх культур, визначати тенденції та закономірності та приймати обґрунтовані рішення щодо зрошення, внесення добрив, боротьби зі шкідниками та інших важливих аспектів управління рослинництвом.

Розробка веб додатку для моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту на основі мови програмування javascript з використанням REACT та бази даних MySQL надасть можливість замінити традиційні, часто малоефективні методи обробки фермерської інформації на сучасну та ефективну автоматизовану систему.

Загалом мета розробки програми полягає, щоб надати фермерам і агрономам комплексну систему моніторингу та управління, яка підвищує

продуктивність, зменшує ризики та сприяє стійким та ефективним методам вирощування сільськогосподарських культур у відкритому грунті.

1.3. Підстави для розробки

Підставою для розробки кваліфікаційної роботи на тему «» є наказ ректора по Національному технічному університету «Дніпровська політехніка» № 350-с від 16.05.2023р. Освітня програма спеціальності 122 «Комп'ютерні науки».

1.4. Постановка завдання

Система моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту, включає наступний вміст:

Завдання проекту – забезпечити фермерів та агрономів надійною системою моніторингу та управління вирощуванням сільськогосподарських культур у відкритому грунті. Техніко-економічна сутність системи полягає в її здатності збирати, аналізувати та інтерпретувати дані для підтримки прийняття обґрунтованих рішень та оптимізації сільськогосподарської практики. Обґрунтування такого рішення випливає з необхідності підвищення продуктивності, зниження ризиків і сприяння стійкості у рослинництві, що призводить до покращення економічних результатів і використання ресурсів.

Розроблена система використовується в управлінні різними об'єктами, включаючи поля відкритого грунту, окремі сільськогосподарські культури, метеостанції, датчики ґрунту, супутникові знімки та технології дистанційного зондування. Він також може охоплювати підрозділи або підприємства, які займаються вирощуванням сільськогосподарських культур у відкритому грунті, наприклад ферми, сільськогосподарські кооперативи або постачальники агрономічних послуг.

Об'єкти інформаційної системи в проекті включають дані про поля (площа,

тип ґрунту, кількість опадів), добрива (азот, фосфор, калій), а також інформацію про культури. Індикатори, що характеризують стан системи, можуть включати показники якості даних, стан калібрування датчика, час безперебійної роботи системи та доступність даних у реальному часі та історичних даних.

Вихідна інформація служить для надання даних у реальному часі та історичних даних про різні параметри, що впливають на виробництво сільськогосподарських культур. Це включає дані про погоду для розуміння умов навколишнього середовища, дані про ґрунт для оцінки наявності вологи та поживних речовин, а також дані про стан рослин для моніторингу росту рослин, виявлення факторів стресу та виявлення шкідників і хвороб.

Система повинна мати вимоги до організації збору даних, включаючи встановлення та обслуговування датчиків, процедури калібрування та протоколи передачі даних. Обробка вхідної інформації включає перевірку даних, агрегацію та аналіз для отримання значущої інформації та практичних рекомендацій. Система також повинна мати положення для контролю та виправлення даних, включаючи механізми обробки аномалій даних або викидів.

Завдання вважається розв’язаним автоматизованим методом, доки не будуть виконані певні умови, такі як завершення збору та обробки даних, генерування думок і рекомендацій, або коли досягнуто певних порогових значень або критеріїв (наприклад, заздалегідь визначена стадія росту культури, рівень зараження шкідниками або вимоги до зрошення).

Система може бути пов’язана з іншими завданнями, такими як планування зрошення, управління поживними речовинами, боротьба зі шкідниками та хворобами та прогнозування врожайності. Інтеграція із зовнішніми системами або платформами, такими як ринкові інформаційні системи або програмне забезпечення для управління фермами, також може покращити процес прийняття рішень і оптимізувати робочі процеси.

Персонал, залучений до вирішення завдань системи, включатиме фермерів, агрономів і техніків, відповідальних за встановлення датчиків, обслуговування та перевірку даних. Технічні засоби включатимуть пристрої

збору даних (метеостанції, ґрунтові датчики), алгоритми обробки та аналізу даних, інтерфейси візуалізації та автоматизовані системи оповіщення. Розподіл функцій змінюватиметься залежно від конкретних ситуацій, при цьому персонал відповідатиме за інтерпретацію даних, прийняття обґрунтованих рішень і виконання рекомендованих дій, тоді як технічні засоби підтримають процеси збору даних, аналізу та звітності.

Розв'язання завдання автоматизованим способом може припинитись у таких випадках:

Виникнення системних помилок або технічних проблем: Якщо комп'ютерна система моніторингу стикається з серйозними технічними проблемами або системними помилками, які не можуть бути вирішені, це може призвести до зупинки роботи системи. Наприклад, апаратна несправність, відмова сенсорів або збої у програмному забезпеченні можуть перешкодити продовженню автоматизованого розв'язання завдання.

Недоцільність використання системи з організаційних або економічних причин: Іноді можуть виникати ситуації, коли використання комп'ютерної системи моніторингу стає недоцільним з організаційних або економічних причин. Наприклад, якщо вартість підтримки і обслуговування системи перевищує отриману користь, організація може вирішити припинити її використання. Також зміни в організаційних процесах або стратегії бізнесу можуть призвести до припинення автоматизованого розв'язання завдання.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Система повинна забезпечувати можливість збору та обробки даних з різних джерел, таких як метеостанції, датчики ґрунту та технології дистанційного зондування.

Пропонувати зручний інтерфейс для введення даних, що дозволяє

користувачам вводити дані структурованим та інтуїтивно зрозумілим способом.

Забезпечувати гнучкі параметри представлення даних, включаючи настроювані звіти, діаграми, графіки та інструменти візуалізації даних.

Підтримувати можливості аналізу даних, такі як аналіз тенденцій, статистичні розрахунки та генерування ключових показників ефективності (KPI), пов’язаних з рослинництвом.

Повинна містити функціональні можливості для створення попереджень, сповіщень і практичних рекомендацій на основі результатів аналізу даних.

1.5.2. Вимоги до інформаційної безпеки

Система повинна забезпечувати доступність, конфіденційність і цілісність даних, захищаючи від несанкціонованого доступу, порушень даних і копіювання програмного забезпечення.

Реалізовувати механізми автентифікації користувачів, контроль доступу та рольові дозволи для контролю доступу до даних і функціональності системи.

Щоб мінімізувати втрату даних і забезпечити безперервність роботи системи, необхідно використовувати процедури регулярного резервного копіювання та аварійного відновлення

Система має відповідати відповідним нормам захисту даних і найкращим галузевим практикам інформаційної безпеки

1.5.3. Вимоги до склада та параметрів технічних засобів

Технічні засоби повинні включати апаратні компоненти, такі як сервери, пристрої зберігання та мережеве обладнання для підтримки вимог до продуктивності та масштабованості системи.

Компоненти програмного забезпечення повинні включати систему управління базою даних, алгоритми обробки даних, модулі інтерфейсу користувача та будь-які необхідні інструменти інтеграції.

Система повинна бути розроблена для обробки очікуваного обсягу даних, забезпечуючи ефективне зберігання, пошук і обробку даних.

Технічні засоби повинні мати достатню обчислювальну потужність і ємність пам'яті для виконання завдань аналізу даних і звітності в прийнятні часові рамки.

Система повинна підтримувати масштабованість і бути здатною пристосуватися до майбутнього зростання та збільшення попиту користувачів.

Додаток повинен мати достатньої ємності жорсткий диск для зберігання даних та резервних копій.

1.5.4. Вимоги до інформаційної та програмної сумісності

Система повинна підтримувати взаємодію з іншими програмними додатками, які зазвичай використовуються в галузі сільського господарства, дозволяючи обмін даними та інтеграцію.

Відповідати стандартним інформаційним структурам і форматам для полегшення обміну даними та сумісності із зовнішніми системами.

Розроблюване програмне забезпечення сумісне з операційними системами Windows XP, 7, 8 чи 10.

Необхідно визначити методи управління вихідним кодом і контролю версій, щоб забезпечити зручність програмного забезпечення та співпрацю між розробниками.

Система має бути розроблена за допомогою мови програмування JavaScript з використанням REACT та MySQL.

РОЗДІЛ 2

ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Система забезпечує можливість автоматизації роботи системи моніторингу даних поля відкритого ґрунту, прискорення та полегшення роботи адміністрації.

Функціональне призначення системи являє собою набір основних функцій, які додаток буде приводити в дію, наприклад перегляд даних поля та ведення бази даних про поля та добрива

Головне вікно запущеної системи моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту показане на рис. 2.1.

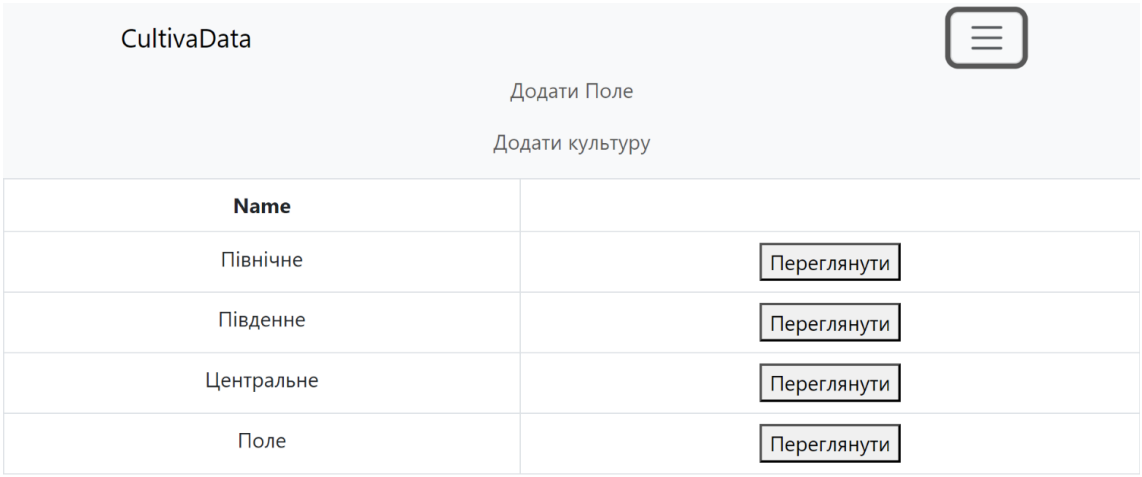

Рис. 2.1. Вікно додатку де перелічені внесені поля

При виборці посилання Додати Поле буде відкрито вікно для внесення даних таких як:

- Назва поля
- Назва культури

• Тип ґрунту

• Очікуваний врожай

Вікно при натисканні Додати Поле показано на рис. 2.2.

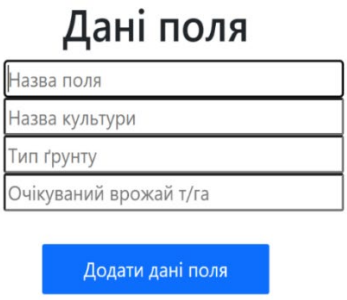

Рис. 2.2. Вікно для внесення даних поля

Далі по внесенню відповідних даних та натисканню кнопки Додати дані поля буде перенесено для вводу даних добрив та виділення поля на карті на рис. 2.3. Після натискання кнопки Додати, данні буде збережено до бази даних а користувача перенесе на головну сторінку Данні для вводу:

• Кислотність поля

• Азот

• Фосфор

• Калій

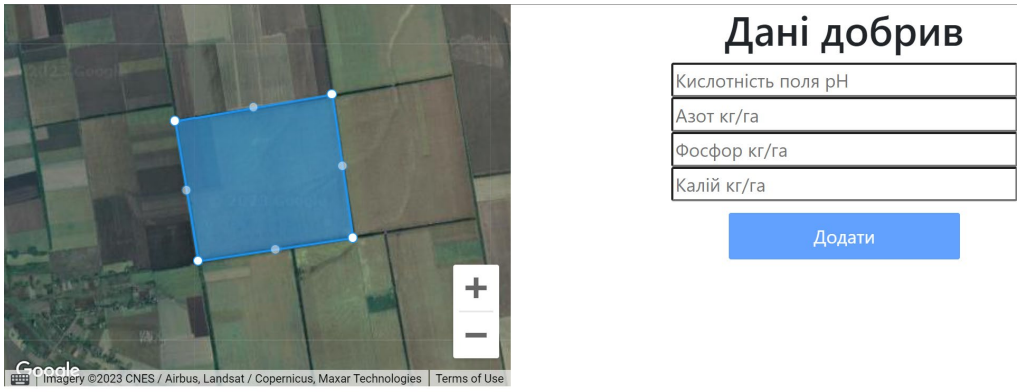

Рис. 2.3. Вікно для внесення даних добрив

Також по кліку на Додати культуру на головній сторінці буде перенесено на сторінку для вводу даних сільськогосподарської культури та виділення поля на карті на рис. 2.4. Дані для вводу:

- Назва культури;
- Мінімальна кислотність;
- Максимальна кислотність.

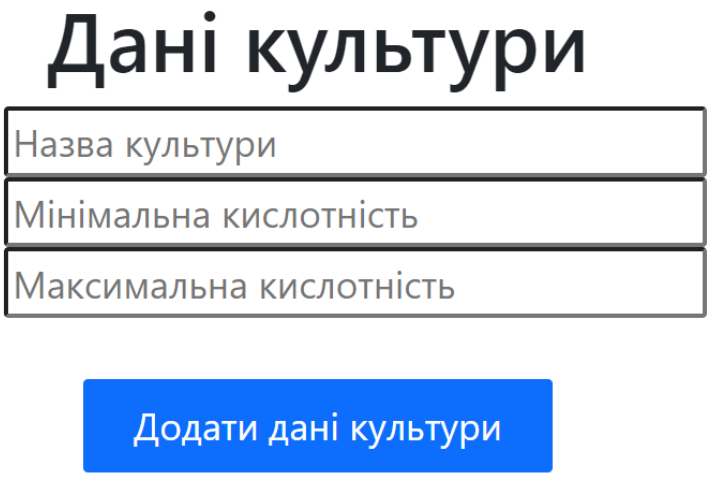

Рис. 2.4. Вікно для внесення даних культури

Розроблений веб додаток для моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту має наступні функціональні можливості.

Управління інформацією про поля: Система дозволяє реєструвати та додавати дані про поля, включаючи назву поля, назву культури, тип ґрунту та очікуваний врожай. Це дозволяє фермерам вести ефективний облік своїх полів та врожаїв.

Управління інформацією про добрива: Веб-додаток надає можливість додавати дані про добрива, такі як кислотність поля, кількість азоту, калію та фосфору. Це допомагає фермерам належним чином планувати та використовувати добрива для оптимального зростання своїх культур.

Управління інформацією про сільськогосподарські культури: Додаток дозволяє додавати дані про різні сільськогосподарські культури, включаючи їх назву, мінімальну та максимальну кислотність. Це допомагає фермерам краще розуміти потреби своїх культур та вживати належних заходів для їх успішного вирощування.

Розроблений веб-додаток для моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту має на меті досягнення таких цілей:

Зниження трудомісткості та підвищення продуктивності: Система дозволяє фермерам автоматизувати ручні робочі процеси та замінити їх автоматизованими, що зменшує трудомісткість та підвищує швидкість виконання завдань. Це сприяє підвищенню продуктивності фермерського господарства та зниженню затрат.

Покращення якості та точності даних: Використання веб-додатка дозволяє уникнути помилок, пов'язаних з ручним введенням та обробкою даних, і забезпечує збереження актуальної та достовірної інформації про поля та сільськогосподарські культури. Це допомагає фермерам приймати обґрунтовані рішення та планувати свою роботу.

Забезпечення зручного та швидкого доступу до інформації: Веб-додаток надає зручний і швидкий доступ до необхідної інформації про поля, добрива та сільськогосподарські культури. Це спрощує роботу фермерів і полегшує обмін інформацією між ними.

Отже, розроблений веб-додаток сприяє покращенню ефективності та точності процесів управління сільськогосподарськими виробництвами рослинництва та забезпечує зручний доступ до важливої інформації для фермерів.

2.2. Опис застосованих математичних методів

У розробці додатку не використовувались математичні методи.

2.3. Опис використаних технологій та мов програмування

Для розробки серверної частини під проект було вирішено використати мову програмування JavaScript, в якості середовища розробки обрана Visual studio code. Використані фреймворки express.js. Бібліотека REACT та об’єктно-реляційна система керування базами даних MySQL.

Вибір цих компонентів ґрунтується на проведених розрахунках і аналізах, які ретельно враховують потреби та вимоги проєкту.

Мова програмування – javascript[1-5].

JavaScript (JS) — це легка, інтерпретована або своєчасно скомпільована мова програмування з першокласними функціями. Хоча він найбільш відомий як мова сценаріїв для веб-сторінок, багато середовищ без браузерів також використовують його, наприклад Node.js, Apache CouchDB і Adobe Acrobat..

JavaScript – це високорівнева мова програмування. Вона підтримує імперативний, функціональний, подієво-орієнтований підходи, має динамічну типізацію та застосовується для запису послідовних операцій — «сценаріїв» чи «скриптів». Такі послідовності зазвичай інтерпретуються, а не компілюються, а тому не потребують додаткових програм чи інструментів перетворення на інший рівень кодування

Сучасний JavaScript є «безпечною» мовою програмування. Він не надає низькорівневий доступ до пам'яті чи ЦП, оскільки спочатку був створений для браузерів, яким це не потрібно.

Широка екосистема бібліотек та фреймворків: JavaScript має велику екосистему, яка включає корисні бібліотеки та фреймворки. Наприклад, Express.js є популярним фреймворком для створення серверних додатків та API на основі Node.js. Це дозволяє легко розробляти серверну частину вашої системи моніторингу та забезпечувати обробку запитів та взаємодію з базою даних.

На відміну від статичних мов програмування, динамічна мова виконує під час виконання багато речей, які робить статична мова під час компіляції. Це має

плюси та мінуси, і це дає нам такі потужні функції, як динамічний тип, пізнє зв’язування, відображення, функціональне програмування, зміна середовища виконання об’єктів, закриття та багато іншого.

Взаємодія з базою даних: JavaScript може використовуватись для взаємодії з реляційною базою даних, такою як MySQL. Це дозволяє вам зберігати, організовувати та аналізувати дані про поля, добрива та культури. Ви можете виконувати запити до бази даних, отримувати, зберігати та оновлювати дані, пов'язані з моніторингом сільськогосподарських виробництв.

Загалом, використання JavaScript у вашій системі моніторингу сільськогосподарських виробництв дозволяє створювати ефективні та інтерактивні веб-інтерфейси, взаємодіяти з зовнішніми сервісами, працювати з базою даних та забезпечувати зручний доступ до необхідної інформації для фермерів.

Фреймворк express.js [6 – 7].

Для розробки серверної в нашому веб додатку ми використовуємо фреймворк express.js.

Express — найпопулярніший веб-фреймворк Node і базова бібліотека для ряду інших популярних веб-фреймворків Node

Express.js є популярним фреймворком для розробки веб-додатків на основі Node.js. Він надає простий та ефективний спосіб побудови серверних додатків з використанням JavaScript. Основна мета Express.js - спростити розробку веб-додатків, забезпечуючи потужні функціональні можливості і гнучкість.

Express.js пропонує простий та зрозумілий інтерфейс, що дозволяє швидко розпочати роботу з фреймворком. Він має мінімальний набір необхідних функцій, що дозволяє розробникам використовувати тільки ті компоненти, які необхідні для їхніх потреб.

Express.js надає потужну систему маршрутизації, яка дозволяє визначати різні шляхи (routes) для обробки різних HTTP-запитів. Це спрощує організацію вашого коду та робить його більш структурованим.

Express.js використовує концепцію middleware, яка дозволяє обробляти HTTP-запити за допомогою серії функцій. Це відкриває безліч можливостей для додаткової обробки запитів, включаючи автентифікацію, журналювання, обробку помилок та багато іншого.

Express.js є дуже гнучким фреймворком, який дозволяє легко розширювати його функціональність за допомогою різних додаткових пакетів. Ви можете використовувати тисячі пакетів, доступних в пакетному менеджері npm, для розширення можливостей додатка.

Підтримка REST API: Express.js є чудовим вибором для створення серверної частини RESTful API. Він надає потужні інструменти для обробки маршрутів, обробки запитів та відповідей, що спрощує створення інтерфейсу API для вашого додатка.

Загалом, Express.js є надійним та широко використовуваним фреймворком, який дозволяє розробникам швидко створювати веб-додатки на основі Node.js. Він надає потужні функції, гнучкість та легкість використання, що робить його відмінним вибором для будь-якого проекту.

База даних – MySQL [8 – 13].

MySQL є однією з найпопулярніших відкритих реляційних баз даних, що використовується у багатьох веб-додатках і системах усього світу. Вона пропонує надійне та швидке зберігання, організацію та керування великими обсягами структурованих даних.

MySQL базується на реляційній моделі, що дозволяє організовувати дані у вигляді таблиць зі зв'язками між ними. Це спрощує структурування та управління даними.

MySQL надає багато вбудованих функцій, таких як операції з вибіркою, сортування, фільтрація, агрегація та об'єднання даних. Вона також підтримує транзакції для забезпечення цілісності та безпеки даних.

MySQL відома своєю швидкодією та ефективністю у роботі з великими обсягами даних. Вона використовує оптимізовані алгоритми запитів та індексування для забезпечення швидкого доступу до даних.

MySQL підтримує горизонтальне та вертикальне масштабування, що дозволяє розширювати базу даних під час зростання обсягів даних та навантаження. Вона також має реплікацію, кластеризацію та інші механізми для розподіленої обробки даних.

MySQL надає різні механізми безпеки, включаючи автентифікацію, авторизацію, шифрування та контроль доступу до даних. Вона також має вбудовані заходи захисту від SQL-ін'єкцій та інших атак на безпеку.

Узагальнюючи, MySQL є потужною та надійною базою даних, яка дозволяє зберігати та керувати даними системи моніторингу сільськогосподарських виробництв рослинництва. Вона забезпечує широкий набір функціональності, швидкодію та безпеку, що робить її відмінним вибором для даного проекту.

Середовище розробки - Visual studio code [14 – 16].

Visual studio code це безкоштовне і легке використання середовище розробки, розроблене компанією Microsoft. Воно стало дуже популярним серед розробників завдяки своїм функціональним можливостям, розширюваності та приємному користувацькому інтерфейсу.

VS Code підтримує операційні системи Windows, macOS і Linux, що робить його доступним для розробників на різних платформах.

VS Code має вбудований термінал, що дозволяє виконувати команди напряму з редактора коду. Це зручно для виконання команд збірки, запуску сервера, встановлення залежностей та інших операцій.

VS Code має можливість відлагоджати JavaScript та код на Node.js прямо в браузері Chrome, що полегшує відлагодження клієнтської та серверної частини веб-додатків.

VS Code надає можливість відлагодження коду прямо у середовищі розробки. Ви можете встановлювати точки зупину, крокувати по коду, переглядати значення змінних та багато іншого, що спрощує відладку програми

Використання Visual Studio Code забезпечує доступ до широкого співтовариства розробників, яке активно підтримує це середовище. Це означає,

що ви можете знайти багато онлайн-ресурсів, форумів, блогів та плагінів, які допоможуть знайти відповіді на будь-які питання, вирішити проблеми та вдосконалити свою роботу з VS Code. Крім того, активна спільнота розробників постійно розширює функціональність VS Code шляхом створення нових розширень та покращення існуючих, що дозволяє використовувати найсучасніші технології та інструменти у вашому процесі розробки.

Загалом, Visual Studio Code є потужним, розширюваним та легким використанням середовищем розробки, яке допомагає розробникам писати, відлагоджувати та керувати своїми проектами зручно та ефективно.

Фронтенд бібліотека – REACT [17-20].

React – це відкрита JavaScript бібліотека для побудови користувацьких інтерфейсів. Вона була розроблена компанією Facebook і використовується для побудови веб-додатків з високою швидкістю та ефективністю.

Основною концепцією React є розбиття інтерфейсу на невеликі ізольовані компоненти, які можуть бути повторно використані. Кожен компонент має свою внутрішню логіку, стан і методи, що дозволяє легко управляти інтерактивними елементами і оновлювати їх стан в залежності від змін.

React використовує віртуальний DOM, що дозволяє ефективно оновлювати лише необхідні частини інтерфейсу при зміні даних. Це забезпечує високу швидкість та ефективність роботи додатків.

React сприяє побудові додатків з використанням компонентної архітектури. Кожен компонент може мати свій власний стан, властивості та методи, що спрощує розробку та підтримку великих проектів.

React рекомендує використовувати односторонній потік даних, що дозволяє прогнозовувати стан компонентів та спрощує управління даними.

React має велику та активну спільноту розробників, яка постійно вносить внески до розширення функціональності бібліотеки, створює корисні пакети, документацію та навчальні матеріали. Це робить React потужним інструментом для розробки веб-додатків

Загалом, React є потужним інструментом для побудови ефективних,

швидких та масштабованих веб-додатків. Його компонентна архітектура, віртуальний DOM та широкі можливості розширення роблять його популярним вибором для розробників у сучасному веб-розробці

Загалом, комбінація JavaScript, React, Express.js, Google API та MySQL забезпечує потужну та гнучку основу для розробки комп'ютерної системи моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту. Вони дозволяють побудувати динамічний інтерфейс, надійний серверний шар, інтеграцію з різними сервісами та ефективне зберігання та обробку даних.

2.4. Опис структури системи та алгоритмів її функціонування

У рамках цього підпункту буде надано детальний опис логічної структури та алгоритмів функціонування веб додатку для моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту. Даний додаток розробляється з метою полегшення та автоматизації процесів, пов'язаних з монітором полів, добрив, та сільськогосподарських культур.

2.4.1. Опис логічної структури системи

Основні компоненти веб додатка для моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту складаються з різних елементів, які взаємодіють між собою, забезпечуючи ефективне та надійне функціонування системи. Основні складові частини системи включають:

Користувацький інтерфейс:

Ця складова частина відповідає за взаємодію з користувачем та забезпечує зручний та інтуїтивно зрозумілий інтерфейс для введення та відображення даних. Вона включає в себе елементи, такі як випадаюче меню, форми, кнопки, таблиці та інші елементи управління, які дозволяють користувачеві взаємодіяти з системою.

Модуль управління даними поля:

Цей модуль відповідає за операції, пов'язані з керуванням даними в полях. Він забезпечує можливість додавання нових даних, редагування існуючих даних та видалення даних. Цей модуль взаємодіє з модулем бази даних, який відповідає за зберігання, отримання та оновлення необхідної інформації, а також її видалення. Інформація відображається через таблицю, яка отримує дані від бази даних.

Модуль управління даними добрив та обчислення площі поля:

Ця компонента відповідає за взаємодію з користувачем і надає зручний та легко зрозумілий інтерфейс для введення та відображення даних. Вона включає різноманітні елементи, такі як випадаючі меню, форми, кнопки, таблиці та інші елементи керування, що дозволяють користувачу взаємодіяти з системою.

Опис алгоритму та функціонування програми:

Для досягнення мети дипломної роботи був розроблений конкретний алгоритм для вирішення поставлених завдань. У процесі розробки веб-додатку для моніторингу відкритого ґрунту сільськогосподарських виробництв рослинництва використовується така схема алгоритму:

Взаємодія з користувачем:

Система взаємодіє з користувачем через користувацький інтерфейс, де він може виконувати різні дії, такі як додавання та редагування даних.

Обробка даних:

Введені користувачем дані підлягають обробці системою відповідно до визначених алгоритмів. Наприклад, при додаванні нової сільськогосподарської культури дані перевіряються на правильність та цілісність перед збереженням у базі даних. Якщо перевірка успішна, система звертається до бази даних і інформація зберігається в ній.

Взаємодія з базою даних:

Система взаємодіє з базою даних для зберігання та отримання інформації. Запити до бази даних виконуються з метою отримання потрібних даних для їх відображення на екрані та подальшої обробки.

Вивід результатів:

Система відображає результати взаємодії з користувачем на екрані, наприклад, шляхом повідомлень про успішне виконання різних дій.

Опис структури бази даних системи:

База даних системи для веб додатку моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту буде включати таблиці, які відображатимуть різні дані. Наприклад, таблиці можуть містити інформацію про користувачів, дані полів, добрив, та культур приклад реалізації наведено на рисунку 2.5.

Рис.2.5 Приклад відображення у додатку занесених елементів до бази даних

Назви полів, культур, добрива та іншу відповідну інформацію. Таблицю даних полів наведено на рис. 2.6. Використання бази даних дозволить зберігати дані в структурованому та доступному форматі, а також забезпечить швидкий доступ до потрібної інформації.

id	name	crop_id	soil_type	gathered
70	Північне	4	Чорнозем	5.000
71	Південне	5	Чорнозем	5.000
72	Центральне	5	Чорнозем	5.000
74	Сонячне	5	Чорнозем	5.000
73	Поле	4	грунт	8.000
NULL	NULL	NULL	NULL	NULL

Рис. 2.6. Таблиця полів

Таблиця з інформацією про сільськогосподарські культури наведено в таблиці 2.7

	id	name	acid_min	acid_max
▶	4	Пшениця	5.500	7.700
	5	Пшоно	4.000	8.000
	6	Гречка	5.000	7.500
*	NULL	NULL	NULL	NULL

Рис 2.7. Таблиці сільськогосподарських культур

Зв'язок системи з іншими програмами:

Наразі розроблений додаток не взаємодіє з іншими програмами, але у перспективі для подальшої розробки можлива реалізація взаємодії.

Як один з можливих розвитків - у разі існування необхідності система може взаємодіяти з іншими програмами, що використовуються в моніторингу полів або з фізичними системи. Наприклад, система може інтегруватись з інтернетом речей, щоб автоматично генерувати дані прямо з поля. Також можлива інтеграція з системами які обчислюють погоду для отримання даних опадів.

Також ймовірна Інтеграція з системою автоматичного поливу, щоб автоматично регулювати полив рослин в залежності від отриманої інформації про вологості ґрунту.

В цілому, описана структура системи та алгоритми її функціонування дають чітке уявлення про компоненти веб-додатку для моніторингу

сільськогосподарських виробництв рослинництва відкритого ґрунту та їх взаємозв'язок, а також процеси обробки даних, взаємодії з користувачем та базою даних.

2.5. Обґрунтування та організація вхідних та вихідних даних програми

При розробці веб-додатка для моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту, належне обґрунтування та організація вхідних та вихідних даних є важливим етапом проектування інформаційної системи. Це включає визначення характеру, організації, попередньої підготовки вхідних даних, а також характеру, організації, формату, опису та способу кодування вихідних даних програми.

2.5.1. Характер, організація і попередня підготовка вхідних даних

Вхідні дані програми для моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту представлені у вигляді цифрових або буквенних символів, які вводяться у текстові поля. Збір інформації включає отримання даних від зовнішніх користувачів, таких як фермери.

Перед введенням даних попередня підготовка включає перевірку їх достовірності, цілісності та відповідності вимогам системи. Наприклад, для координатних даних може бути здійснена обробка, що обмежує кількість знаків після коми до п'яти. Це сприяє забезпеченню відповідності формату та точності даних, що вводяться.

2.5.2. Характер і організація вихідних даних

Вихідні дані програми включають результати обробки вхідних даних та відповідні відображення на екрані користувача. Характер вихідних даних

подається у різноманітному вигляді.

До вхідних даних проекту належать наступні дані, які записуються до бази даних

Таблиця 2.1

Вхідні об’єкти

Метод	Дані	Опис
addData	id	Id поля
	name	Назва поля
	crop_id	Id культури
	soil_type	Тип ґрунту
	gathered	Очікуваний врожай
addMap	id	Id запису
	id_field	Номер поля
	acidity	Кислотність поля
	nitrogen	Азот
	phosphorus	Фосфор
	potassium	Калій
	changed_time	Остання дата створення/зміни
	latlng	Координати поля
	square	Площа поля
addCrop	id	Id культури
	name	Назва культури
	acidity_min	Мінімальна кислотність

	acidity_max	Максимальна кислотність
addUser	id	Id культури
	user	Логін користувача
	password	Пароль

Організація вихідних даних включає підготовку результатів обробки вхідних даних для подальшого відображення або збереження. Наприклад, результати можуть бути структуровані відповідно до потреб користувача та відображені у вигляді зрозумілих списків.

До вихідних даних належать всі дані, що виводяться з бази даних під час роботи проекту:

Таблиця 2.2

Вихідні об’єкти

Метод	Дані	Опис
getAllMaps	id	Id поля
	name	Назва поля
	crop_id	Id культури
	soil_type	Тип ґрунту
	gathered	Очікуваний врожай
getMapsInfo	id	Id запису
	id_field	Номер поля
	acidity	Кислотність поля
	nitrogen	Азот
	phosphorus	Фосфор
	potassium	Калій

	changed_time	Остання дата створення/зміни
	latlng	Координати поля
	square	Площа поля
getCropInfo	id	Id культури
	name	Назва культури
	acidity_min	Мінімальна кислотність
	acidity_max	Максимальна кислотність
getAllUsers	id	Id культури
	user	Логін користувача
	password	Пароль

2.5.3. Формат, опис і спосіб кодування вхідних та вихідних даних

Формат, опис і спосіб кодування вхідних та вихідних даних мають вирішальне значення для ефективного обміну інформацією та сумісності з іншими системами. Вхідні дані повинні бути структуровані і організовані для подальшої обробки, а їх формат може бути текстовим, XML, JSON або іншим, залежно від потреб системи. Опис вхідних та вихідних даних включає структуру, типи полів та їх значення, а також правила валідації.

Наприклад, опис структури даних сільськогосподарської культури може містити поля, такі як назва культури, мінімальна кислотність, максимальна кислотність. Спосіб кодування визначається з урахуванням ефективності передачі та обробки даних. Зазвичай використовують стандартні кодування, наприклад, UTF-8, для підтримки різних мов та символів.

Обґрунтування та організація вхідних та вихідних даних дозволяють забезпечити ефективний обмін інформацією, точність даних та сумісність з іншими системами. Це важливий аспект розробки веб-додатку для моніторингу

сільськогосподарських виробництв рослинництва відкритого ґрунту, що забезпечує його ефективну та надійну роботу.

2.6. Опис розробленої системи

2.6.1. Використані технічні засоби

З урахуванням сучасного інформаційного середовища та вимог до функціональності додатку було проведено відбір оптимальних технічних рішень, які гарантують ефективність та надійність системи.

Для реалізації веб додатку були використані наступні технічні засоби:

Комп'ютерна система:

Процесор: Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz 2.50 GHz з тактовою частотою 2.5 ГГц та 6 ядрами, що забезпечує потужні обчислювальні можливості та швидку обробку даних.

Оперативна пам'ять: 8 ГБ DDR3 з тактовою частотою 3000 МГц, що дозволяє отримати оптимально швидкий доступ до даних та підтримувати багатозадачність.

Жорсткий диск: SSD ємністю 500 ГБ, що забезпечує швидкий та необмежений доступ до файлів та даних додатку.

Відеокарта: NVIDIA GeForce GTX 1650 з 4 ГБ відеопам'яті, що дозволяє забезпечити оптимальну продуктивність графічного інтерфейсу додатку.

Операційна система:

Microsoft Windows 10: популярна операційна система. Має інтуїтивний інтерфейс та розширені функції. Підтримує різне апаратне забезпечення. Надає широкі можливості та забезпечує безпеку.

Даний набір технічних засобів був вибраний з метою забезпечення оптимальної продуктивності та ефективності роботи веб-додатка для моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту.

Ці технічні ресурси об'єднують обчислювальні можливості, стабільність операційної системи, зручність розробки та надійне зберігання даних. В результаті, система моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту може працювати з високою продуктивністю та розширеними функціональними можливостями.

2.6.2. Використані програмні засоби

Для успішної розробки та оптимального функціонування веб-додатку для моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту, були використані передові технічні рішення. В цьому підпункті будуть наведені докладні характеристики використаних технічних засобів, включаючи типи електронних обчислювальних машин і пристроїв.

Комп'ютери: Операційна система: Для розробки та тестування веб-додатку використовувалися потужні персональні комп'ютери, які відповідали вимогам розробки. Оперативна пам'ять була належної ємності, а обчислювальна потужність та швидкодія комп'ютерів забезпечували ефективну роботу засобів розробки та тестування.

Операційна система: Для розробки та експлуатації додатку використовувалася оновлена версія операційної системи - Windows 10. Ця операційна система забезпечила зручне та стабільне робоче середовище, підтримку сучасних технологій та розробницьких інструментів.

Інтегроване середовище розробки (IDE): Для розробки додатку та написання програмного коду використовувалось потужне та популярне інтегроване середовище розробки (IDE) - Visual Studio Code. Це середовище надало широкі можливості для програмування на мові JavaScript, зокрема підсвічування синтаксису, автоматичне завершення коду та інструменти для налагодження.

База даних: Для зберігання та управління даними системи моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту було

використано реляційну базу даних MySQL. Ця база даних забезпечила надійну та ефективну структуру для зберігання різних типів інформації, таких як поля, добрива, культури та інші важливі дані.

Мова програмування та фреймворк: У процесі розробки графічного інтерфейсу користувача була використана мова програмування Javascript. Ця мова є дуже популярною серед веб-розробників та має широкий вибір інструментів і бібліотек для розробки графічних інтерфейсів. Для створення фронтенду була використана бібліотека REACT, яка дозволила створити естетичний та функціональний користувацький інтерфейс з великими можливостями налаштування.

Використання цих технічних засобів дало можливість успішно розробити та забезпечити ефективне функціонування веб-додатку для моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту. Вони забезпечили зручність у процесі розробки, надійність та ефективність системи, що відповідає вимогам власників ферм та їх користувачів.

2.6.3. Виклик та завантаження програми

Першим кроком у виклику додатку є визначення оптимального методу його активації. Для забезпечення зручності користувачів та ефективного моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту, передбачається розробка інтуїтивно зрозумілого інтерфейсу, який може бути запущений за допомогою пошуку в пошуковику за URL: https://CultivaData.

При розгляді процесу завантаження веб-додатку варто звернути увагу на два основні аспекти: вимоги до обсягу оперативної пам'яті та розмір самого додатку. З метою забезпечення надійної роботи додатку та підтримки розмаїття функціональності, був проведений докладний аналіз вимог до оперативної пам'яті. Враховуючи оптимальні рекомендації, були встановлені мінімальні та рекомендовані значення обсягу оперативної пам'яті, які є необхідними для

стабільного функціонування додатку.

Крім того, розмір додатку є важливим фактором при його завантаженні. З метою забезпечення швидкого доступу та ефективної роботи були проведені оптимізаційні заходи, спрямовані на зменшення обсягу додатку, без погіршення його функціональності та якості роботи.

За рахунок оптимізації обсягу веб-додатка досягнуто швидкого завантаження та ефективного використання дискового простору.

В підсумок, в розділі "Виклик та завантаження додатку" у нашій дипломній роботі, яка зосереджена на розробці веб-додатка для моніторингу відкритих сільськогосподарських виробництв рослинництва, ми розглянули важливі аспекти, такі як вибір оптимального методу доступу до додатку, відповідність системним вимогам, вимоги до оперативної пам'яті та оптимізацію обсягу програми. Ці критичні питання сприятимуть зручності використання додатку, його швидкому завантаженню та надійній роботі, допомагаючи впровадити ефективний та сучасний інструмент для оптимізації моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту.

2.6.4. Опис інтерфейсу користувача

Під час запуску додатку, користувач бачить перед собою сторінку авторизації, на якій є поля для вводу логіну та пароля, кнопка «Увійти» а також посилання «Зареєструватись» яке відправить на сторінку реєстрації. З даної сторінки можна потрапити на головну сторінку додатку увівши правильні логін та пароль. Сторінку авторизації додатку наведено на рис. 2.8.

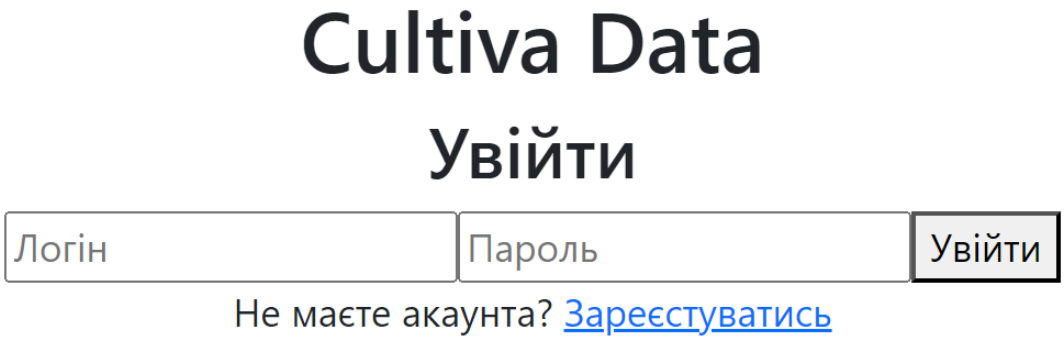

Рис. 2.8. Сторінка аторизації додатку

Якщо користувач натиснув на посилання «Зареєструватись», то він побачить перед собою сторінку створення користувача дуже схожу на попередню. На ній є поля для вводу логіну та пароля, кнопка «Зареєструватись» а також посилання «Увійти» яке відправить на сторінку входу. З даної сторінки можна потрапити на головну сторінку додатку та зберегти свої дані увівши свої логін та пароль. Сторінку авторизації додатку наведено на рис. 2.9.

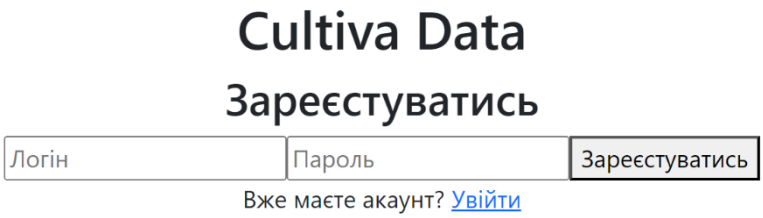

Рис. 2.9. Сторінка аторизації додатку

Далі, користувач попадає на головну сторінку, де відображаються:

- Меню з лого додатку та посиланнями «Додати Поле» і «Додати культуру»
- Список полів з наступним змістом:
- Назва поля

- Кнопка «Переглянути»

Головну сторінку наведено на рис. 2.10.

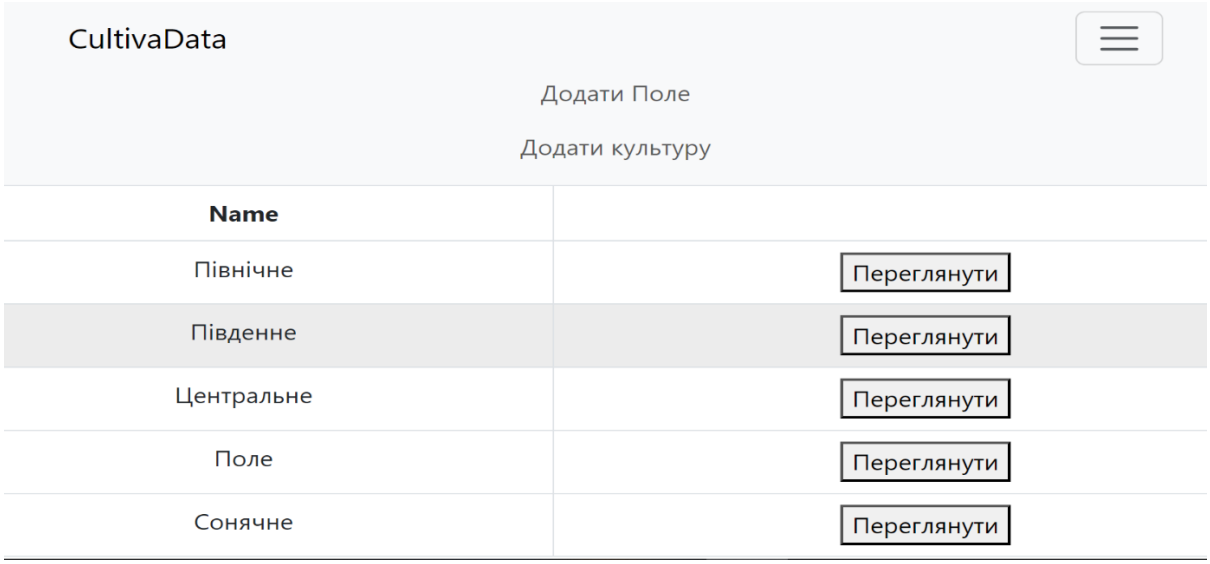

Рис. 2.10. Головна сторінка

Якщо користувач натисне на одну з кнопок «Переглянути». Для нього згенерується модальне вікно з наступними данними:

- Карта з google.api, на якій згенеровано полігон поля з кольором кислотності
- Дані добрив:
  - Азоту
  - Фосфору
  - Калію
- Дані поля:
  - Назва поля
  - Тип ґрунту
  - Очікуваний врожай
  - Площа поля

- Рівні кислотності для заданої культури

  - Рівні кислотності для заданої культури
  - Дата створення
  - Кнопка «Закрити»

Модальне вікно наведено на рис. 2.11.

Рис. 2.11. Модальне вікно

РОЗДІЛ 3

ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки інформаційної системи

Вихідні дані для розробки програмного забезпечення мають такі характеристики:

1) Передбачуване число операторів: 1496.
2) Коефіцієнт складності програми: 1,5.
3) Коефіцієнт кореляції програми під час розробки: 0,4.
4) Середня годинна заробітна плата програміста: 120 грн/год
5) Вартість однієї машино-години ЕОМ: 15 грн/год.

3.1 Розрахунок трудомісткості та вартості розробки програмного продукту

Ускладнення процесу нормування праці під час створення програмного забезпечення пояснюється творчим характером роботи програміста. Внаслідок цього, для оцінки трудомісткості розробки ПЗ може бути використана система моделей з різним рівнем точності.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{ , людино-годин}$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C(1 + p)$$

q – передбачуване число операторів;

C – коефіцієнт складності програми;

p – коефіцієнт кореляції програми в ході її розробки.

$$Q = 1496 \times 1{,}5(1 + 0{,}4) = 3142;$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q * B}{(75 \ldots 85)K}, \text{ людино-годин.}$$

де *B* – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі, *B*=1,3;

K – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності становить 1,35;

$$t_u = \frac{3142*1,3}{80*1,35} = 68,93, \text{людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t€ = \frac{Q}{(20...25)K}$$

$$t_a = \frac{3142}{20*1,35} = 212.1, \text{людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K}$$

$$t_n = \frac{3142}{22*1,35} = 192.81, \text{людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)K}$$

$$t_n = \frac{3142}{5*1,35} = 848,34, \text{людино-годин.}$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису

$$tЉр = \frac{Q}{(15 \ldots 20)K}$$

$$t_{др} = \frac{3142}{17*1,35} = 249.51, \text{людино-годин.}$$

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації

$$tЉ” = 0,75 \cdot tЉр$$

$t_{до} = 0,75 \cdot 249.51 = 187.1$, людино-годин

$t_д = 249,51 + 187,1 = 436.61$, людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення: t = 50+68,93+212,1+192,81+848.34+436,61=1808.79 , людино-годин.

У результаті ми розрахували, що в загальній складності необхідно 1808.79 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $K_{по}$ включають витрати на заробітну плату

виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$\text{жпм} = ,,\text{п} + ,\text{Ме, грн}$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$,,\text{п} = t \cdot ,,\text{пт, грн,}$$ де t – загальна трудомісткість, людино-годин;

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година

$З_{ЗП} = 1808.79 * 120 = 217054{,}8$, грн.

$З_{МВ}$– Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$\text{ЗМВ} = t\text{отл} \cdot \text{СМ, грн,}$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год. $С_{МЧ}$ – вартість машино-години ЕОМ, грн/год.

$З_{МВ} = 848{,}34 \cdot 15 = 12725$, грн.

$К_{ПО} = 217054{,}8 + 12725 = 229779{,}8$ грн.

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ мес.}$$

де B_k - число виконавців;

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

$$T = \frac{1808.79}{1 \cdot 176} = 10{,}28 \text{ міс.}$$

Висновки. Час розробки даного програмного забезпечення складає 1808.79 людино-годин. Таким чином, очікувана тривалість розробки складе 10,28 місяця при 40 годинному робочому тижні (місячний фонд робочого часу 176 годин), а витрати на створення програмного забезпечення складатимуть 229779,8 грн.

ВИСНОВКИ

Розроблений додаток для моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту є незамінним інструментом для оптимізації робочих процесів та управління даними. Цей додаток надає зручний інтерфейс для фермерів, дозволяючи їм ефективно вести облік полів, культур і інших важливих деталей. Використання такого додатку сприяє уникненню помилок, забезпечує швидкий доступ до необхідної інформації та спрощує процеси прийняття рішень.

Застосування цього додатку може покращити точність та надійність обробки даних, що є незмінно важливим для забезпечення високої якості роботи в аграрній сфері. Впровадження цього додатку дозволить ефективно організувати робочий процес та спростити завдання фермерів. Це сприятиме забезпеченню ефективної роботи в аграрному секторі та полегшить їхні завдання.

Оцінка одержаних результатів відносно аналогів:

Розроблений додаток для моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту має значні переваги по відношенню до аналогів. Він пропонує зручний інтерфейс користувача, реалізований з використанням бібліотеки REACT, що спрощує взаємодію з системою для фермерів.

Досягнутий ступінь новизни:

Додаток є інноваційним завдяки використанню сучасних технологій та інструментів. Його комбінація надає зручний та ефективний інструмент для фермерів, який відповідає сучасним вимогам.

Практичне та наукове значення результатів:

Розроблений додаток має практичне значення, оскільки його використання допомагає автоматизувати рутинні процеси в моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту. Він спрощує облік полів, добрив та культур, що дозволяє фермерам ефективно

виконувати ці завдання та сконцентруватись на більш складних та важливих аспектах їхньої роботи. Такий додаток покращує продуктивність та сприяє оптимальному управлінню сільськогосподарськими виробництвами..

Прогнозні припущення про подальший розвиток:

Додаток для моніторингу сільськогосподарських виробництв рослинництва відкритого ґрунту пропонує безліч можливостей для постійного вдосконалення та розширення. Наприклад, в майбутньому можна впровадити функціонал автоматичної генерації звітів та статистики, розширити масштаби роботи з більшим обсягом даних та вдосконалити інтерфейс користувача за допомогою сучасних технологій та привабливого дизайну. Крім того, можливе розширення підтримки різних типів користувачів, що забезпечить більш широке використання додатку та його адаптацію до індивідуальних потреб користувачів. Всі ці можливості сприятимуть забезпеченню сталого розвитку додатку та задоволенню зростаючих потреб користувачів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи 1808.79 люд-год, проведений підрахунок вартості роботи по створенню програми 229779,8 грн. та розраховано час на його створення 10 місяців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація JavaScript URL:https://developer.mozilla.org/en-Us/docs/Web/ JavaScript (Дата звернення: 11.05.2023)
2. Розробка на JavaScript URL: https://dou.ua/lenta/articles/how-to-learn-javascript/ (Дата звернення: 11.05.2023)
3. Офіційна документація JavaScript URL:https://developer.mozilla.org/uk/docs/Web/JavaScript/Guide (Дата звернення: 12.05.2023)
4. The complete JavaSctipt Handbook URL:https:/medium.com/free-code-camp/the-complete-javasctipt-handbook-f26b2c171719c (Дата звернення: 12.05.2023)
5. Основи використання npm для управління пакетами у проєктах JavaScript URL:https://www.sitepoint.com/npm-guide/ (Дата звернення: 25.05.2023)
6. Повне керівництво з вивчення Express.js URL:http//developer.mozilla.org/uk/uk/docs/Learn/Server-side/Express_Node.js (Дата звернення: 12.05.2023)
7. Розробка веб-додатків з використанням Express.js та MySQL URL: https://bezkoder.com/node-js-express-sequelize-mysql/ (Дата звернення: 12.05.2023)
8. Навчальний посібник з використання MySQL в JavaScript URL: https://www.mysqltutorial.org/mysql-nodejs/ (Дата звернення: 25.05.2023)
9. Використання MySQL Workbench для управління базами даних URL: https://dev.mysql.com/doc/workbench/en/ (Дата звернення: 25.05.2023)
10. Офіційна документація MySQL URL: https://dev.mysql.com/doc/ (Дата звернення: 25.05.2023)
11. Підручник з використання MySQL URL: https://www.mysqltutorial.org/ (Дата звернення: 13.05.2023)

12. Підключення до MySQL бази даних за допомогою Node.js та пакету mysql /URL: https://www.sitepoint.com/using-node-mysql-javascript-client/ (Дата звернення: 20.05.2023)

13. Основи MySQL з простими прикладами URL: https://www.mysqltutorial.org/basic-mysql-tutorial.aspx (Дата звернення:20.05.2023)

14. Використання Visual Studio Code для розробки Javascript та React проєктів URL: https://code.visualstudio.com/docs/languages/javascript (Дата звернення: 20.05.2023)

15. Офіційна документація Visual Studio Code URL:https://code.visualstudio.com/docs (Дата звернення: 20.05.2023)

16. Посібник з використання Visual Studio URL: https://www.freecodecamp.org/news/visual-studio-code-beginner-tutorial-tips/ (Дата звернення: 25.05.2023)

17. Початковий навчальний посібник з React URL: https://reactjs.org/tutorial/tutorial.html (Дата звернення: 13.05.2023)

18. Офіційна документація React / URL:https://uk.reactjs.org/docs/getting-started.html (Дата звернення: 13.05.2023)

19. Вступ React URL:https://uk.reactjs.org/tutorial/tutorial.html (Дата звернення: 13.05.2023)

20. Розробка веб додатків з використанням React та Express URL: https://bezkoder.com/react-express-js-rest-api-example/ (Дата звернення: 25.05.2023)

ДОДАТОК А

КОД ПРОГРАМИ

mapController.js

```
const database = require("../database"); module.exports.addData = (req, res) => { const name = req.body.name; const crop_id = req.body.crop_id; const soil_type = req.body.soil_type; const gathered = req.body.gathered; const checkName = "SELECT name FROM fields WHERE name=?"; return new Promise((resolve, reject) => { database.query(checkName, [name], (err, response) => { if (err) { reject(err); return; } if (response && response.length > 0) { console.log(response); res.status(422).json({ message: "This place is already added" }); } else { database.query( "INSERT INTO fields(name, crop_id, soil_type, gathered) VALUES(?, ?, ?,?)", [name, crop_id, soil_type, gathered], (err, result) => { if (err) { reject(err); return; } if (result) { res.status(200).json({ message: "Поле збережено" }); } else { res.status(400).json({ message: "Something went wrong" }); } } ); } resolve(response); }); }); }; module.exports.mapInfo = (req, res) => { const name = req.body.name; const sql = "SELECT * FROM fields WHERE name=?"; return new Promise((resolve, reject) => { database.query(sql, name, (err, result) => { if (err) { reject(err); return; } if (result) { res.status(200).json({ result }); } else { res.status(400).json({ message: "Something went wrong" }); } resolve(result); }); }); }; module.exports.addMap = (req, res) => { const fieldId = req.body.fieldId; const acidity = req.body.acidity; const nitrogen = req.body.nitrogen; const phosphorus = req.body.phosphorus; const potassium = req.body.potassium; const coordinates = req.body.coordinates; const square = req.body.square; const query = "INSERT INTO points (id_field, acidity, nitrogen, phosphorus, potassium, latlng, square) VALUES (?,?,?,?,?,?,?) "; return new Promise((resolve, reject) => { database.query( query, [fieldId, acidity, nitrogen, phosphorus, potassium, coordinates, square], (err, result) => { if (err) { reject(err); return; } if (result) { res.status(200).json({ message: "Дані збережено" }); } else { res.status(400).json({ message: "Something went wrong" });
```

```
} resolve(result); } ); }); }; module.exports.getAllMaps = (req, res) => { const selectsql = "SELECT * FROM fields"; return new Promise((resolve, reject) => { database.query(selectsql, (err, result) => { if (err) { reject(err); } if (result) { res.status(200).json(result); } if (err) { res.status(400).json({ msg: "Something went wrong" }); } resolve(result); }); }); }; module.exports.getMapsInfo = (req, res) => { const id_field = req.body.fieldId; const sql = "SELECT * FROM points WHERE id_field=?"; return new Promise((resolve, reject) => { database.query(sql, id_field, (err, result) => { if (err) { reject(err); } if (result) { res.status(200).json(result); } resolve(result); }); }); }; module.exports.getCropInfo = (req, res) => { const id = req.query.crop_id; const sql = "SELECT * FROM crops WHERE id=?"; return new Promise((resolve, reject) => { database.query(sql, [id], (err, result) => { if (err) { reject(err); return; } if (result) { res.status(200).json({ result }); } else { res.status(400).json({ message: "Something went wrong" }); } resolve(result); }); }); }; module.exports.getCropByName = (req, res) => { const name = req.query.crop_name; const sql = "SELECT * FROM crops WHERE name=?"; return new Promise((resolve, reject) => { database.query(sql, [name], (err, result) => { if (err) { reject(err); return; } if (result) { res.status(200).json({ result }); } else { res.status(400).json({ message: "Something went wrong" }); } resolve(result); }); }); }; module.exports.addCrop = (req, res) => { const name = req.body.name; const acid_min = req.body.acid_min; const acid_max = req.body.acid_max; const checkName = "SELECT name FROM crops WHERE name=?"; return new Promise((resolve, reject) => { database.query(checkName, [name], (err, response) => { if (err) { reject(err); return; } if (response && response.length > 0) { console.log(response); res.status(422).json({ message: "Така культура вже існує" }); } else { database.query( "INSERT INTO crops(name, acid_min, acid_max) VALUES(?, ?, ?)", [name, acid_min, acid_max], (err, result) => { if (err) { reject(err); return; } if (result) { res.status(200).json({ message: "Культуру додано" }); } else { res.status(400).json({ message: "Something went wrong" }); } } ); } resolve(response); }); }); };
```

userController.js

```
const database = require("../database"); module.exports.getAllUsers = (req, res) => { const selectsql = "SELECT * FROM users"; return new Promise((resolve, reject) => { database.query(selectsql, (err, result) => { if (err) { reject(err); } if (result) { res.status(200).json(result); } if (err) { res.status(400).json({ msg: "Something went wrong" }); } resolve(result); }); }); }; module.exports.addUser = (req, res) => { const user = req.body.user; const password = req.body.password; const query = "INSERT INTO users (user, password) VALUES (?,?) "; return new Promise((resolve, reject) => { database.query(query, [user, password], (err, result) => { if (err) { reject(err); return; } if (result) { res.status(200).json({ message: "Дані збереженo" }); } else { res.status(400).json({ message: "Something went wrong" }); } resolve(result); }); }); };
```

mapRoute.js

```
const express = require("express"); const { addData, mapInfo, addMap, getAllMaps, getMapsInfo, getCropInfo, addCrop, getCropByName } = require("../controllers/mapController"); const { getAllUsers, addUser } = require("../controllers/userController"); const router = express.Router(); router.post('/addData', addData) router.post('/getMapInfo', mapInfo) router.post('/addMap',addMap) router.get('/getAllMaps',getAllMaps) router.post('/allMapsInfo', getMapsInfo) router.get('/getCropInfo',getCropInfo) router.post('/addCrop', addCrop) router.get('/getCropByName', getCropByName) router.get('/getAllUsers', getAllUsers) router.post('/addUser', addUser) module.exports = router
```

app.js

```
const express = require("express"); const app = express(); const env = require("dotenv"); const cors = require("cors"); const database = require("./database"); const router = require("./routes/mapRoute"); env.config(); database app.use(express.json()); app.use(cors()) app.use("/api", router) const PORT = process.env.PORT; app.listen(PORT, () => console.log(`Server running on PORT ${PORT}`));
```

database.js

```
require("dotenv").config(); const mysql = require("mysql2") const connect = mysql.createConnection({ host: process.env.DATABASE_HOST, user: process.env.DATABASE_USER, password: process.env.DATABASE_PASSWORD, database: process.env.DATABASE_NAME }) connect.connect(function(err){ if(err){ throw err; } else{ console.log("Connect successful") } }) module.exports = connect;
```

addCrops.js

```
import React, { useState } from "react"; import { Button } from "react-bootstrap"; // import { useNavigate  } from 'react-router-dom' import axios from "axios"; import { useHistory } from "react-router-dom"; export default function AddCrop() { //     const navigate = useNavigate (") const history = useHistory(); const [name, setName] = useState(); const [acid_min, setAcid_min] = useState(); const [acid_max, setAcid_max] = useState(); let message = ""; const addCropData = () => { axios .post("http://localhost:2000/api/addCrop", { name: name, acid_min: acid_min, acid_max: acid_max, }) .then((response) => { const { message } = response.data; alert(message); history.push("/"); }) .catch((error) => { if (error.response && error.response.status === 422) { const { message } = error.response.data; alert(message); } else { console.log(error); } }); }; return ( <div> <div style={{ marginTop: "10px" }}></div> <br /> <h1>Дані культури</h1> <ul> <li> <input type="text" placeholder="Назва культури" onChange={(e) => setName(e.target.value)} value={name} className="searchtext" /> </li> <li> <input type="text" placeholder="Мінімальна кислотність" onChange={(e) => setAcid_min(e.target.value)} value={acid_min} className="searchtext" /> </li> <li> <input type="text" placeholder="Максимальна кислотність" onChange={(e) => setAcid_max(e.target.value)} value={acid_max} className="searchtext" /> </li> </ul> <Button className="searchbutton" disabled={name == "" ? true : false} onClick={addCropData} > Додати дані культури </Button> </div> ); }
```

addField.js

```
import React, { useEffect, useRef, useState } from "react"; import { useParams, useHistory } from "react-router-dom"; import { Button } from "react-bootstrap"; import axios from "axios"; import Field from "./Field"; function AddField() { let btnRef = useRef(); let { name } = useParams(); const [mapLocation, setLocation] = useState(); const history = useHistory(); const [mapInfo, setMapInfo] = useState([]); const [acidity, setAcidity] = useState(""); const [nitrogen, setNitrogen] = useState(""); const [phosphorus, setPhosphorus] = useState(""); const [potassium, setPotassium] = useState(""); const [area, setArea] = useState(); const handleAreaUpdate = (newArea) => { setArea(newArea); }; useEffect(() => { async function fetchData() { try { const response = await fetch( `https://maps.googleapis.com/maps/api/geocode/json?address=Dnipro&key=${proce ss.env.REACT_APP_API_KEY}` ); const data = await response.json(); if (data.results && data.results.length > 0) { const { lat, lng } = data.results[0].geometry.location; if (isFinite(lat) && isFinite(lng)) { const truncatedLat = Number(lat.toFixed(5)); // Truncate to 6 decimal places const truncatedLng = Number(lng.toFixed(5)); // Truncate to 6 decimal places setLocation({ lat: truncatedLat, lng: truncatedLng }); } } axios .post("http://localhost:2000/api/getMapInfo", { name: name }) .then((response) => { const mapData = response.data.result[0]; setMapInfo(mapData); }) .catch((error) => { console.error(error); }); } catch (error) { console.error(error); } } fetchData(); return () => { // setLocation(null) // setMapInfo(null) }; }, [name, mapInfo.crop_id]); const [state, setState] = useState([]); const { paths } = state; const new_path = JSON.stringify(state.paths); const saveMap = async () => { const fieldId = mapInfo.id; await axios .post("http://localhost:2000/api/addMap", { fieldId, acidity, nitrogen, phosphorus, potassium, coordinates: new_path, square: area, }) .then((response) => { if (response) { alert(`${response.data.message}`); history.push("/"); } else { alert("Something went wrong"); } }) .catch((error) => { console.log(error); }); }; return ( <div id="container"> <Field apiKey={process.env.REACT_APP_API_KEY} center={mapLocation} paths={paths} point={(paths) => setState({ paths })}
```

updateArea={handleAreaUpdate} /> <div style={{ marginTop: "10px" }}>
   <h1>Дані добрив</h1>   <input type="text" placeholder="Кислотність поля pH" onChange={(e) => setAcidity(e.target.value)} value={acidity} className="searchtext" />   <input type="text" placeholder="Азот кг/га" onChange={(e) => setNitrogen(e.target.value)} value={nitrogen} className="searchtext" />   <input type="text" placeholder="Фосфор кг/га" onChange={(e) => setPhosphorus(e.target.value)} value={phosphorus} className="searchtext" />   <input type="text" placeholder="Калій кг/га" onChange={(e) => setPotassium(e.target.value)} value={potassium} className="searchtext" />   <Button className="searchbutton" disabled={ !paths || paths.length < 1 || acidity == "" || nitrogen == "" || phosphorus == "" || potassium == "" ? true : false } onClick={saveMap} > Додати </Button>   ); } export default AddField;

AddFieldData.js

import React, { useState } from "react"; import { Button } from "react-bootstrap"; // import { useNavigate } from 'react-router-dom' import axios from "axios"; import { useHistory } from "react-router-dom"; export default function AddFieldData() { // const navigate = useNavigate (") const history = useHistory(); const [name, setName] = useState(); const [crop_name, setCrop_name] = useState(); const [soil_type, setSoilType] = useState(); const [gathered, setGathered] = useState(); let message = ""; const addData = async () => { try { const response = await axios.get(`http://localhost:2000/api/getCropByName?crop_name=${crop_name}`); const cropId = JSON.parse(response.data.result[0].id); const postData = { name: name, crop_id: cropId, soil_type: soil_type, gathered: gathered, }; const postResponse = await axios.post("http://localhost:2000/api/addData", postData); const { message } = postResponse.data; alert(message); history.push(`/map/${name}`); } catch (error) { if (error.response && error.response.status === 422) { const { message } = error.response.data; alert(message); } else { console.log(error); } } }; return (<div style={{ marginTop: "10px" }}>
 <h1>Дані поля</h1>

<input type="text" placeholder="Назва поля" onChange={(e) => setName(e.target.value)} value={name} className="searchtext" />   <input type="text" placeholder="Назва культури" onChange={(e) => setCrop_name(e.target.value)} value={crop_name} className="searchtext" />   <input type="text" placeholder="Тип ґрунту" onChange={(e) => setSoilType(e.target.value)} value={soil_type} className="searchtext" />   <input type="text" placeholder="Очікуваний врожай т/га" onChange={(e) => setGathered(e.target.value)} value={gathered} className="searchtext" />   <Button className="searchbutton" disabled={name == "" ? true : false} onClick={addData} > Додати дані поля </Button>); }

AuthSevice.js

import axios from "axios"; class AuthService { async login(username, password) { let signinStatus; await axios .get("http://localhost:2000/api/getAllUsers") .then((response) => { if (response) { const user = response.data.find((element) => element.user === username); const pass = response.data.find((element) => element.password === password); if (user) { if (pass) { alert("signin successful"); signinStatus = true; } else { signinStatus = false; } } else { signinStatus = false; } } else { alert("Something went wrong"); } }) .catch((error) => { console.log(error); }); return signinStatus; } async createUser(userName, password) { let signinStatus; const user = userName; await axios .post("http://localhost:2000/api/addUser", { user, password, }) .then((response) => { if (response) { alert(`${response.data.message}`); signinStatus = true; } else { alert("Something went wrong"); signinStatus = false; } }) .catch((error) => { console.log(error); }); console.log(signinStatus); return signinStatus; } } export default new AuthService();

Field.js

import React, { useCallback, useRef, useState } from "react"; import { GoogleMap, LoadScript, DrawingManager, Polygon, } from "@react-google-maps/api"; export default function Field({ apiKey, center, paths = [], point, updateArea, }) { const [path, setPath] = useState(paths); const [state, setState] = useState({ drawingMode:

"polygon", }); const libraries = ["drawing"]; const mapOptions = { streetViewControl: false, fullscreenControl: false, mapTypeId: "hybrid", mapTypeControl: false, }; const options = { drawingControl: true, drawingControlOptions: { drawingMode: ["Polygon"], }, polygonOptions: { fillColor: "#2196F3", strokeColor: "#2196F3", fillOpacity: 0.5, strokeWeight: 2, clickable: true, editable: true, draggable: true, zindex: 1, }, }; function calculatePolygonArea(paths) { const numPoints = paths.length; let area = 0; for (let i = 0; i < numPoints; i++) { const currentPoint = paths[i]; const nextPoint = paths[(i + 1) % numPoints]; area += convertToRadian(nextPoint.lng - currentPoint.lng) * (2 + Math.sin(convertToRadian(currentPoint.lat)) + Math.sin(convertToRadian(nextPoint.lat))); // area += (currentPoint.lat * nextPoint.lng) - (nextPoint.lat * currentPoint.lng); } area = (area * 6378137 * 6378137) / 2; // Convert the calculated area from square degrees to square kilometers return Math.abs(area / 1e6); } function convertToRadian(input) { return (input * Math.PI) / 180; } const onPolygonComplete = React.useCallback(function onPolygonComplete(poly) { const polyArray = poly.getPath().getArray(); let paths = []; let latArray = []; let lngArray = []; polyArray.forEach(function (path) { paths.push({ lat: Number(path.lat().toFixed(5)), lng: Number(path.lng().toFixed(5)), }); }); const newArea = calculatePolygonArea(paths); updateArea(newArea); console.log(newArea); setPath(paths); point(paths); poly.setMap(null); }, [point]); const polygonRef = useRef(null); const listenerRef = useRef([]); const onEdit = useCallback(() => { if (polygonRef.current) { const nextPath = polygonRef.current .getPath() .getArray() .map((latLng) => { return { lat: latLng.lat(), lng: latLng.lng() }; }); setPath(nextPath); point(nextPath); } }, [setPath, point]); const onLoad = useCallback((polygon) => { polygonRef.current = polygon; const path = polygon.getPath(); listenerRef.current.push(path.addListener("set_at", onEdit), path.addListener("insert_at", onEdit), path.addListener("remove_at", onEdit)); }, [onEdit]); const onUnmount = useCallback(() => { listenerRef.current.forEach((lis) => lis.remove()); polygonRef.current = null; }, []); return (<LoadScript id="script-loader" googleMapsApiKey={apiKey} libraries={libraries}

language="en" region="us" > <GoogleMap mapContainerClassName="appmap" center={center} zoom={12} options={mapOptions} > {path.length === 0 || path.length === 1 || path.length === 2 ? (<DrawingManager drawingMode={state.drawingMode} options={options} editable draggable onPolygonComplete={onPolygonComplete} onMouseUp={onEdit} onDragEnd={onEdit} />) : (<Polygon options={{ fillColor: "#2196F3", strokeColor: "#2196F3", fillOpacity: 0.5, strokeWeight: 2, }} editable path={path} onLoad={onLoad} onUnmount={onUnmount} onMouseUp={onEdit} onDragEnd={onEdit} />)} </GoogleMap> </LoadScript>); }

Header.js

import React from "react"; import { Navbar, Nav, Container } from "react-bootstrap"; function Header() { return (<header> <Navbar bg="light" expand="lg"> <Container> <Navbar.Brand href="/">CultivaData</Navbar.Brand> <Navbar.Toggle aria-controls="basic-navbar-nav"></Navbar.Toggle> <Navbar.Collapse id="basic-navbar-nav"> <Nav className="ms-auto"> <Nav.Link href="/addFieldData">Додати Поле</Nav.Link> <Nav.Link href="/addCrop">Додати культуру</Nav.Link> </Nav> </Navbar.Collapse> </Container> </Navbar> </header>); } export default Header;

Home.js

import React, { useEffect, useState } from "react"; import { Table } from "react-bootstrap"; // import { useNavigate } from 'react-router-dom' import axios from "axios"; import Map from "./Map"; import Header from "./Header"; export default function Home() { const [allMaps, setAllMaps] = useState([]); let message = ""; const getAllMaps = () => { axios .get("http://localhost:2000/api/getAllMaps") .then((response) => { setAllMaps(response.data); }) .catch((error) => { if (error.response && error.response.status === 422) { const { message } = error.response.data; alert(message); } else { console.log(error); } }); }; useEffect(() => { let unmounted = false; setTimeout(() => { getAllMaps(); }, 1000); return () => { unmounted = true; setAllMaps([]); }; }, []); return (<Header /> <Table

bordered hover> <thead> <tr> <th>Name</th> </tr> </thead> <tbody> {allMaps.map((map) => (<Map key={map.id} id={map.id} name={map.name} map={map} />))} </tbody> </Table>); }

Map.js

import React, { useEffect, useState } from "react"; import Modal from "react-modal"; import axios from "axios"; import ViewMap from "./ViewMap"; export default function Map({ id, name, map }) { const [modalView, setModalView] = useState(false); const [cropInfo, setCropInfo] = useState([]); const [acidity, setAcidity] = useState(); const [color, setColor] = useState(); const [acidityValues, setAcidityValues] = useState([]); const [fertilizerAreaData, setFertilizerAreaData] = useState(); const openViewModal = () => { setModalView(true); }; const [coordinates, setCoordinates] = useState(); // const [allMaps, setAllMaps] = useState([]); const getAllMaps = () => { axios .get(`http://localhost:2000/api/getCropInfo?crop_id=${map.crop_id}`) .then((response) => { setCropInfo(response.data.result[0]); // setAllMaps(response.data); }) .catch((error) => { if (error.response && error.response.status === 422) { const { message } = error.response.data; alert(message); } else { console.log(error); } }); }; useEffect(() => { let unmounted = false; setTimeout(() => { if (!unmounted) return getAllMaps(); }, 50); return () => { unmounted = true; }; }, []); useEffect(() => { let unmounted = false; setTimeout(() => { if (!unmounted) { axios .post("http://localhost:2000/api/allMapsInfo", { fieldId: map.id }) .then((response) => { if (response.data.length > 0 && response.data[0].latlng) { setFertilizerAreaData(response.data[0]); setCoordinates(JSON.parse(response.data[0].latlng)); setAcidity(JSON.parse(response.data[0].acidity)); } }); } }, 100); return () => { unmounted = true; }; }, [map.id]); let points = []; if (coordinates !== undefined) { for (let i = 0; i < coordinates.length; i++) { const { lat, lng } = coordinates[i]; if (isFinite(lat) && isFinite(lng)) { const truncatedLat = Number(lat.toFixed(5)); // Truncate to 2 decimal places const truncatedLng = Number(lng.toFixed(5)); // Truncate to 2 decimal places points.push({ lat: truncatedLat, lng: truncatedLng }); } }

```
} const [state, setState] = useState(); const getColorForAcidity = (acidity) => { const numberMax = Number(cropInfo.acid_max); const numberMin = Number(cropInfo.acid_min); const range = (numberMax - numberMin) / 3; setAcidityValues([ numberMin, Number((numberMin + range).toFixed(3)), Number((numberMin + 2 * range).toFixed(3)), numberMax, ]); if (acidity < numberMin) { return "#00008B"; } else if (acidity < numberMin + range) { return "#0000FF"; } else if (acidity < numberMin + 2 * range) { return "#00FF00"; } else if (acidity <= numberMax) { return "#FFA500"; } else if (acidity > numberMax) { return "#FF0000"; } }; // Function to calculate the color based on the acidity value useEffect(() => { let unmounted = false; setTimeout(() => { if (!unmounted) { let colorTemp = getColorForAcidity(acidity); setColor(colorTemp); } }, 300); return () => { unmounted = true; }; }, [acidity]); return ( <> <tr> <td>{name}</td> <td> <button onClick={openViewModal}>Переглянути</button> </td> </tr> <Modal isOpen={modalView} ariaHideApp={false} contentLabel="View Map"> <ViewMap points={points} setModalView={setModalView} color={color} fertilizerAreaData={fertilizerAreaData} cropInfo={cropInfo} acidityValues={acidityValues} map={map} /> </Modal> </> ); }
```

SignInPage.js

```
import React, { useState } from "react"; import AuthService from "./AuthService"; import { useHistory } from "react-router-dom"; const SignInPage = () => { const history = useHistory(); const [userName, setUserName] = useState(""); const [password, setPassword] = useState(""); const [message, setMessage] = useState(""); const [authenticationStatus, SetAuthenticationStatus] = useState(false); if (authenticationStatus) { history.push("/home"); } const handleSignIn = async (e) => { e.preventDefault(); try { const authenticated = await AuthService.login(userName, password); if (authenticated) { SetAuthenticationStatus(true); history.push("/home"); } else { setMessage("Неправильний логін або пароль"); } } catch (error) { console.log("Error signing in:", error.message); } }; return ( <div className="container"> <header> <h1>Cultiva Data</h1> </header> <main> <div className="auth-container"> <h2>Увійти</h2> {message && <p>{message}</p>}
```

<form onSubmit={handleSignIn}> <input type="text" name="userName" placeholder="Логін" value={userName} onChange={(e) => setUserName(e.target.value)} required /> <input type="password" name="password" placeholder="Пароль" value={password} onChange={(e) => setPassword(e.target.value)} required /> <button type="submit">Увійти</button> </form> Не маєте акаунта? Зареєструватись   </main>); }; export default SignInPage;

SignUpPage.js

import React, { useState } from "react"; import AuthService from "./AuthService"; import { useHistory } from "react-router-dom/cjs/react-router-dom.min"; const SignUpPage = () => { const history = useHistory(); const [userName, setUserName] = useState(""); const [password, setPassword] = useState(""); const [message, setMessage] = useState(""); const [authenticationStatus, SetAuthenticationStatus] = useState(false); if (authenticationStatus) { history.push("/home"); } const handleSignUp = async (e) => { e.preventDefault(); try { const authenticated = await AuthService.login(userName, password); if (!authenticated) { try { const userCreated = await AuthService.createUser(userName, password); if (userCreated) { SetAuthenticationStatus(true); history.push("/home"); } else { setMessage("Error"); } } catch (error) { console.log("Error signing up:", error.message); } } else { setMessage("User already exists"); } } catch (error) { console.log("Error signing up:", error.message); } }; return (<header> <h1>Cultiva Data</h1> </header> <main> <h2>Зареєструватись</h2> {message && {message} } <form onSubmit={handleSignUp}> <input type="text" name="userName" placeholder="Логін" value={userName} onChange={(e) => setUserName(e.target.value)} required /> <input type="password" name="password" placeholder="Пароль" value={password} onChange={(e) => setPassword(e.target.value)} required /> <button type="submit">Зареєструватись</button> </form>

Вже маєте акаунт? Увійти   </main>); }; export default SignUpPage;

ViewMap.js

import React from "react"; import { GoogleMap, LoadScript, Polygon } from "@react-google-maps/api"; function ViewMap({ points, setModalView, color, fertilizerAreaData, cropInfo, acidityValues, map, }) { const mapOptions = { streetViewControl: false, fullscreenControl: false, mapTypeId: "hybrid", mapTypeControl: false, }; console.log(fertilizerAreaData); return (<LoadScript id="script-loader" googleMapsApiKey={process.env.REACT_APP_API_KEY} language="en" region="us" > {points.length > 1 ? (<GoogleMap options={mapOptions} mapContainerClassName="viewmap" center={points[0]} zoom={13} > <Polygon path={points} options={{ fillColor: color, strokeColor: "#2196F3", fillOpacity: 0.5, strokeWeight: 2, clickable: true, editable: false, draggable: false, }} /> </GoogleMap>) : null} </LoadScript> Дата створення {fertilizerAreaData.changed_time.split("T")[0]}   <h4>Рівні кислотності грунту для культури: {cropInfo.name}</h4> Колір тексту вказує на колір полігону на карті   <p style={{ color: "#00008B" }}> 0 - {cropInfo.acid_min} pH дуже низька кислотність   <p style={{ color: "#0000FF" }}> {" "} {cropInfo.acid_min} - {acidityValues[1]} pH низька кислотність   <p style={{ color: "#00FF00" }}> {" "} {acidityValues[1]} - {acidityValues[2]} pH оптимальна кислотність   <p style={{ color: "#FFA500" }}> {" "} {acidityValues[2]} - {cropInfo.acid_max} pH висока кислотність   <p style={{ color: "#FF0000" }}> {" "} {cropInfo.acid_max} pH дуже висока кислотність   <h5>Дані поля {map.name}</h5>   Тип ґрунту: {map.soil_type}   Очікуваний врожай: {map.gathered} т/га   Площа поля {fertilizerAreaData.square} кв.км   <h4>Дані добрив</h4>   Азоту - {fertilizerAreaData.nitrogen} кг/га   Фосфору -

{fertilizerAreaData.phosphorus} кг/га   Калію - {fertilizerAreaData.potassium} кг/га   <button onClick={() => setModalView(false)}>Закрити</button>); } export default ViewMap;

App.css

#container { flex: 1; display: flex; justify-content: space-between; } #field-data { height: fit-content; width: fit-content; align-self: flex-start; background-color: #f5f5f5; padding: 10px; } body { margin: 0; padding: 0; } .App { text-align: center; } .searchtext { width: 300px; height: 30px; border-radius: 2px; } .searchbutton { width: 200px; height: 40px; border-radius: 2px; margin-top: 10px; text-align: center; } .appmap { height: 80vh; width: 50vw; } ul { list-style-type: none; } .viewmap { height: 80vh; width: 40vw; }

App.js

import './App.css'; import {BrowserRouter as Router, Switch, Route} from "react-router-dom" import Home from './Components/Home'; import AddField from './Components/AddField'; import AddFieldData from './Components/AddFieldData'; import AddCrop from './Components/AddCrop'; import SignInPage from './Components/SignInPage'; import SignUpPage from './Components/SignUpPage'; function App() { return (<Router> <Switch> <Route exact path="/home" component={Home}/> <Route exact path="/" component={SignInPage}/> <Route exact path="/signup" component={SignUpPage}/> <Route exact path="/addFieldData" component={AddFieldData}/> <Route exact path="/addCrop" component={AddCrop}/> <Route exact path="/map/:name" component={AddField}/> </Switch> </Router>); } export default App;

ДОДАТОК Б
ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В
ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_Лєпшаков.doc	Пояснювальна записка до дипломного проекту. Документ Word.
Диплом_Лєпшаков.pdf	Пояснювальна записка до дипломного проекту в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми
Додаткові файли	
Model.pkt	Файл демонстраційної моделі
Презентація	
Презентація_Лєпшаков.ppt	Презентація дипломного проекту