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 *Галети Андрія Юрійовича*
(ПІБ)

академічної групи *121-19-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка веб-інтерфейсу віддаленого моніторингу системи крапельного поливу*

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	*проф. Лактіонов І.С.*			
розділів:				
спеціальний	*проф. Лактіонов І.С.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*ст.викл. Мартиненко А.А.*			

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 *121-19-1* (група) *Галета Андрія Юрійовича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-інтерфейсу віддаленого моніторингу системи крапельного поливу*

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***	***13.05.2023 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 *Галета А.Ю,.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *12.06.2023р.*

РЕФЕРАТ

Пояснювальна записка: 92 с., 14 рис., 1 табл., 3 дод., 20 джерел.

Об'єкт дослідження – веб-додаток для комп'ютера і смартфона, що призначений для віддаленого моніторингу системи крапельного поливу.

Мета кваліфікаційної роботи – розробка функціонального веб-інтерфейсу, що дозволить віддалено моніторити та керувати системою крапельного поливу. Робота має на меті створення зручного і легкодоступного інструменту для фермерів або власників садів, щоб вони могли в реальному часі відстежувати стан поливної системи, контролювати рівень вологості ґрунту та забезпечувати оптимальні умови поливу рослин.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформи для розробки, виконано проектування і розробка програми, описана робота програми, алгоритм і структура її функціонування, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні веб-додатка, що надає можливість власникам садів або фермерам у реальному часі відстежувати стан своєї системи крапельного поливу з будь-якого місця. Інтерфейс надасть їм інформацію про рівень вологості ґрунту, стан роботи системи, можливі проблеми чи аварії. Це дозволить їм оперативно реагувати на зміни у поливній системі та уникнути можливих проблем.

Актуальність інформаційної системи визначається попитом на крапельний полив, який є більш екологічною альтернативою інших методів поливу, таких як звичайний полив або зрошування. Цей метод дозволяє зменшити випаровування води, уникнути втрат через вітер, а також забезпечити більш точне та цілеспрямоване використання водних ресурсів.

Ключові слова: ІНФОРМАЦІЙНА СИСТЕМА, КОМП’ЮТЕР, ОБЛІК, БАЗА ДАНИХ, АВТОМАТИЗАЦІЯ, ПРОЕКТУВАННЯ, ДОДАТОК, МЕНЮ.

ABSTRACT

Explanatory note: 92 pages, 14 pictures, 1 table, 3 appendix, 20 sources.

The object of research is a web application for a computer and a smartphone designed to remotely monitor a drip irrigation system.

The purpose of the qualification work is to develop a functional web interface that will allow remote monitoring and control of a drip irrigation system. The work aims to create a convenient and easily accessible tool for farmers or garden owners so that they can monitor the status of the irrigation system in real time, control the level of soil moisture and ensure optimal watering conditions for plants.

In the introduction, the analysis and current state of the problem is considered, the purpose of the qualification work and the field of its application are specified, the justification of the relevance of the topic is given, and the statement of the task is clarified.

In the first section, the subject area is analyzed, the relevance of the task and the purpose of the development is determined, the task statement is formulated, and the requirements for software implementation, technologies and software tools are specified.

In the second section, available solutions are analysed, platforms for development are selected, program design and development is performed, program operation, algorithm and structure of its functioning are described, as well as program calling and loading, input and output data are determined, and the composition of technical means parameters is characterized.

In the economic section, the labour intensity of the developed information systemis determined, the cost of work on creating the program is calculated, and the time forits creation is calculated.

The practical significance is to create a web application that allows garden owners or farmers to monitor the status of their drip irrigation system in real time from any location. The interface will provide them with information about the level of soil moisture, the state of the system, possible problems or accidents. This will allow them to respond quickly to changes in the irrigation system and avoid possible problems.

The relevance of the information system is determined by the demand for drip irrigation, which is a more environmentally friendly alternative to other irrigation methods, such as conventional watering or irrigation. This method reduces water evaporation, avoids wind losses, and ensures more accurate and targeted use of water resources.

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

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних;

ІС - інформаційна система;

ПЗ – програмне забезпечення;

ІТ – інформаційні технології.

ЗМІСТ

ВСТУП

За останні два десятиліття сільськогосподарські системи отримали вигоду від впровадження технологічних досягнень, розроблених для інших галузей, таких як GPS, системи зв'язку та системи візуалізації. З урахуванням цього, моя кваліфікаційна робота має на меті розробку та впровадження веб-додатку, який дозволяє здійснювати віддалений моніторинг та управління системою крапельного зрошення.

Зв'язок проблеми, що вирішується, з об'єктом діяльності фахівця, пов'язаного з інформаційними технологіями, проявляється у потребі отримувати та обробляти дані, які надходять від датчиків та пристроїв на полях. Оскільки спеціаліст займається розробкою веб-інтерфейсу для віддаленого моніторингу системи крапельного поливу, важливо мати можливість отримувати актуальну інформацію про стан поливної системи, наприклад, рівень вологості ґрунту, температуру тощо. Для цього необхідно налагодити зв'язок з датчиками на полі та забезпечити їхню інтеграцію з розробленим веб-інтерфейсом. Таким чином, фахівець використовує свої знання в галузі інформаційних технологій для отримання, обробки та відображення даних від датчиків, що дозволяє ефективно моніторити та керувати системою крапельного поливу.

Метою кваліфікаційної роботи є розробка веб-інтерфейсу для віддаленого моніторингу системи крапельного поливу, що базується на мові програмування JavaScript. Цей веб-інтерфейс дозволить користувачам відстежувати та керувати параметрами поливу з будь-якого пристрою, підключеного до Інтернету, забезпечуючи зручну та ефективну систему моніторингу.

Актуальність теми спрямована на ефективне використання водних ресурсів. У сучасних умовах, коли проблема доступності води стає все більш актуальною, важливо шукати способи її ефективного використання. Системи крапельного поливу дозволяють подавати воду безпосередньо до кореневої зони рослин, уникнувши втрат води через випаровування та розсіювання.

Постановка завдання кваліфікаційної роботи полягає в розробці зручного, безпечного та масштабованого веб-інтерфейсу, який надасть можливість ефективного контролю та управління системою поливу для збереження водних ресурсів та підвищення ефективності поливної системи. Сам веб-інтерфейс буде реалізований за допомогою web технологій: Html [1], CSS [2], JavaScript [3].

В якості бази даних для зберігання даних з датчиків була обрана БД – MongoDB [4, 5]. Вона дозволяє розподілити дані на кілька серверів (кластерів), що дозволяє розширювати потужність бази даних та забезпечувати швидкий доступ до інформації. Це важливо для системи крапельного поливу, оскільки можуть виникати потреби в обробці великого обсягу даних в реальному часі.

MongoDB не вимагає жорсткої схеми даних, що дозволяє додавати, видаляти та змінювати поля в документах без необхідності перетворення всієї бази даних. Це особливо корисно для системи крапельного поливу, де можуть з'являтися нові типи датчиків чи параметрів поливу.

Вбудована реплікація та балансування навантаження: MongoDB забезпечує можливість створення реплік бази даних, що дозволяє забезпечити збереження та доступність даних навіть у випадку відмови одного сервера. Також існує можливість автоматичного балансування навантаження між серверами для оптимального розподілу даних та запитів.

Таким чином, ця кваліфікаційна робота націлена а розробку веб-інтерфейсу для віддаленого моніторингу системи крапельного поливу. Основна мета полягає в тому, щоб забезпечити зручний та ефективний спосіб контролю та управління поливною системою за допомогою веб-інтерфейсу.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Сільськогосподарські системи поливу востаннє десятиліття отримали значну користь від використання сучасних інформаційних технологій. Однак, існують проблеми, пов'язані з ефективним моніторингом та управлінням системами поливу, зокрема системами крапельного поливу [6].

Розробка веб-інтерфейсу для віддаленого моніторингу системи крапельного поливу стає актуальною в контексті постійного розвитку інформаційних технологій та їх впровадження у сільськогосподарські галузі. Такий веб-інтерфейс надає можливість оперативно отримувати дані про стан поливної системи, контролювати та налагоджувати її параметри, а також аналізувати та зберігати відповідну інформацію.

Основою для розробки продукту був аналіз вже існуючих сайтів зарубіжних сайтів який надають послуги віддаленого моніторингу системи крапельного поливу.

Ось деякі з них:

1) https://www.rainmachine.com/

Сайт "RainMachine" пропонує широкий вибір продуктів для крапельного поливу з сучасним дизайном, детальними описами та інтеграцією з різними платформами для зручного управління системами поливу.

Головною його особливістю є те, що сайт надає демо-версію свого веб-інтерфейсу. Де любий користувач може ознайомитися з інтерфейсом і функціоналом, який безпосередньо стосується моніторингу системи крапельного поливу. Головна сторінка моніторингу системи крапельного поливу зображена на рис. 1.1.

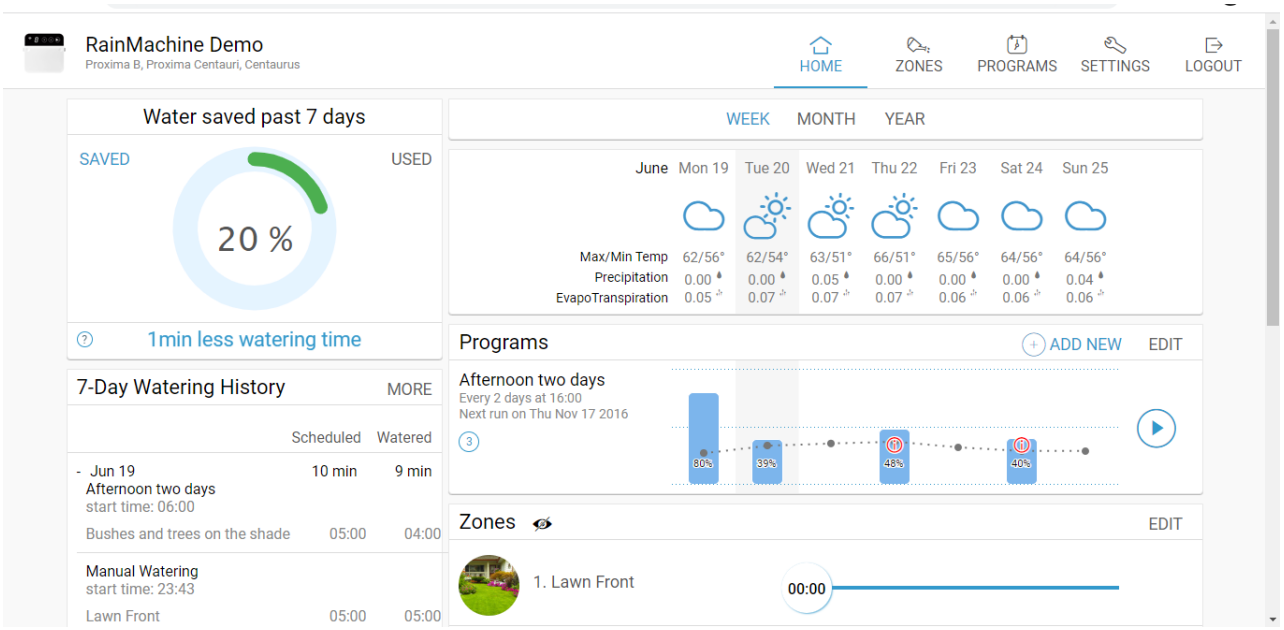

Рис. 1.1. Головна сторінка моніторингу поливу

Веб-інтерфейс має сучасний та чистий дизайн, що сприяє зручному використанню. Кольорова палітра і типографіка добре підібрані, створюючи приємний візуальний досвід.

Меню та розділи розташовані логічно і легко доступні для користувача. Зручність використання покращується завдяки інтуїтивно зрозумілим піктограмам та кнопкам.

Веб-інтерфейс дозволяє користувачам моніторити та керувати системою крапельного поливу. За допомогою графіків, графіків і статистики можна отримати детальний огляд процесу поливу, включаючи історію поливу, часові розклади та статус системи. Інтерактивні елементи дозволяють користувачам змінювати налаштування поливу, додавати нові розклади і контролювати інтенсивність поливу для кожної зони.

Сам сайт адаптивний, що означає, що він добре відображається на різних пристроях, включаючи комп'ютери, планшети та смартфони. Це забезпечує зручний доступ до моніторингу системи крапельного поливу навіть на мобільних пристроях.

З врахуванням важливості безпеки даних, веб-інтерфейс RainMachine забезпечує захист інформації користувачів, використовуючи шифрування та

заходи безпеки.

2) https://www.hydrawise.com/

Сайт "Hydrawis"» є платформою для моніторингу та керування системами крапельного поливу. Він пропонує зручний веб-інтерфейс зі сучасним дизайном, який дозволяє користувачам налаштовувати розклади поливу, контролювати інтенсивність поливу та отримувати статистику. Забезпечуючи функціональність та зручність використання, hydrawise.com створює зручні умови для ефективного моніторингу системи крапельного поливу. Головна моніторингу поливу представлена на рис. 1.2.

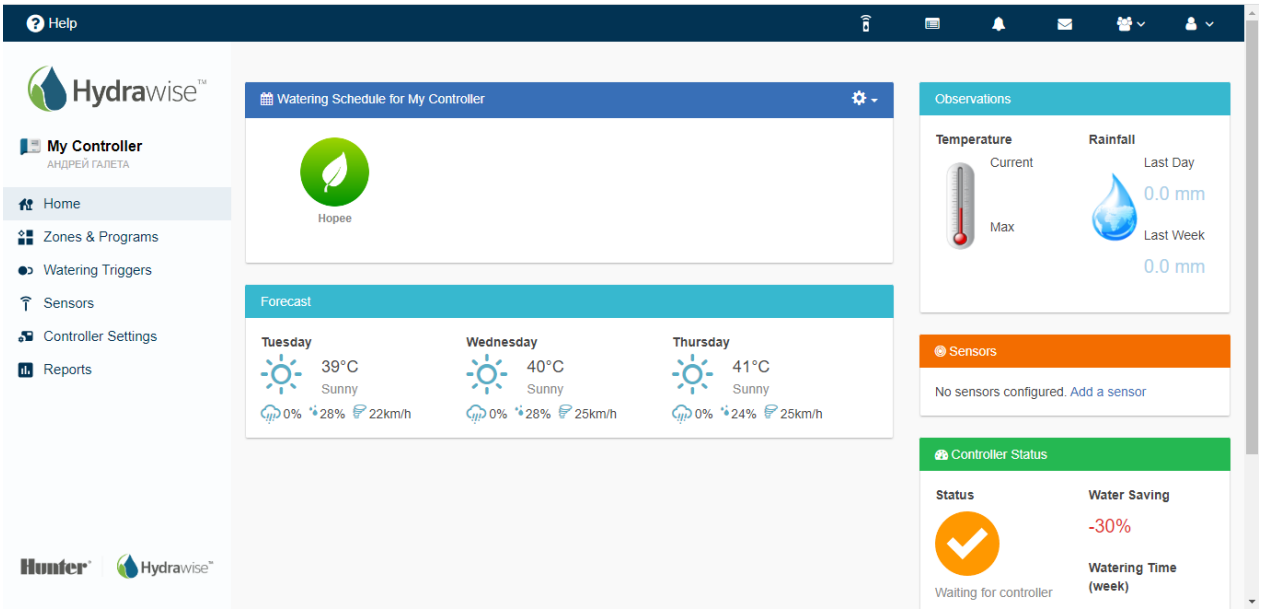

Рис. 1.2. Головна сторінка моніторингу системи крапельного поливу.

Веб-інтерфейс має сучасний та естетичний дизайн, з чистим і привабливим виглядом. Використання приємних кольорів, зручного розташування елементів та інтуїтивно зрозумілих піктограм допомагає створити приємний візуальний досвід для користувача.

Сайт дозволяє користувачам моніторити та керувати системою крапельного поливу. Користувачі можуть налаштовувати розклади поливу, контролювати інтенсивність поливу для кожної зони, переглядати статистику та графіки, а також отримувати сповіщення про стан системи.

Меню та розділи сайту розташовані логічно і добре структуровані, що дозволяє користувачам легко знаходити потрібну інформацію та функції. Меню навігації знаходиться на верхній частині сторінки, що забезпечує зручний доступ до основних функцій.

Узагальнюючи, веб-інтерфейс на сторінці має зручний та естетичний дизайн, пропонує багато функціональних можливостей для моніторингу та керування системою крапельного поливу, і забезпечує зручний доступ до необхідної інформації для користувачів.

Під час аналізу можуть бути виявлені наступні технічні протиріччя, такі як масштабованість: якщо проект передбачає масштабування для використання великим числом користувачів або систем поливу, можуть виникати проблеми зі швидкодією та потужністю. Недостатньо потужна інфраструктура або неефективна оптимізація можуть стати причиною затримок та недостатньої продуктивності. Швидкість та продуктивність: веб-інтерфейс повинен працювати швидко та ефективно, забезпечуючи миттєву відповідь на дії користувача. Недостатня продуктивність, довгі часи завантаження або затримки можуть негативно вплинути на користувачів та їхню взаємодію з системою.

Виявлення прогалин в знаннях та навичках полягає у визначенні основних недоліків, що виникають під час використання наявних програм для віддаленого моніторингу системи крапельного поливу. Аналізується досвід користувачів та їх проблеми, що виникають при використанні програмного забезпечення, а також розглядаються можливі шляхи вирішення цих проблем.

Нездійснені вимоги до виробів або розробок на тему "Розробка веб-інтерфейсу віддаленого моніторингу системи крапельного поливу" можуть включати фізичні обмеження, наприклад, якщо система крапельного поливу встановлена на великій відстані від веб-інтерфейсу, можуть виникати проблеми зі зв'язком та передачею даних, що може ускладнити віддалений моніторинг. Деякі вимоги до розробки веб-інтерфейсу можуть бути нездійснені через обмежений бюджет. Наприклад, реалізація деяких додаткових функцій чи інтеграція зі сторонніми системами може вимагати великих фінансових витрат,

які не доступні у рамках проекту. Або через технічні обмеження, такі як обмежена пам'ять або потужність обчислювальних пристроїв, недостатня швидкість Інтернет-з'єднання або несумісність з наявною інфраструктурою.

1.2. Призначення розробки та галузь застосування

Розробка веб-інтерфейсу віддаленого моніторингу системи крапельного поливу має на меті забезпечити зручний та ефективний спосіб контролю та управління системою поливу з віддаленого місця. Цей веб-інтерфейс надає можливість власникам або операторам сільськогосподарських угідь відстежувати та керувати режимами поливу безпосередньо зі свого комп'ютера, планшета або смартфона.

Галузь застосування цієї розробки включає сільське господарство, садівництво, ландшафтний дизайн та інші області, де використовується система крапельного поливу. Власники сільськогосподарських угідь, городників та ландшафтних дизайнерів використовують цю технологію для ефективного та економічного зрошення своїх культур, квітників та зелених насаджень.

Застосування веб-інтерфейсу для віддаленого моніторингу системи крапельного поливу дозволяє користувачам здійснювати налаштування параметрів поливу, контролювати рівень вологості ґрунту, відстежувати статистику поливу, а також отримувати повідомлення та сповіщення про стан системи. Це спрощує процес управління поливом, покращує використання водних ресурсів, забезпечує оптимальні умови росту рослин і сприяє збільшенню врожайності та здоров'я рослин.

Таким чином, веб-інтерфейс для віддаленого моніторингу системи крапельного поливу є інноваційним інструментом, який сприяє автоматизації та удосконаленню процесу поливу. Він дозволяє ефективно використовувати ресурси, забезпечувати оптимальні умови для рослин та підвищувати продуктивність в галузях, де полив має важливе значення.

1.3. Підстави для розробки

Підставою для розробки кваліфікаційної роботи на тему «Розробка веб-інтерфейсу віддаленого моніторингу системи крапельного поливу» є наказ ректора по Національному технічному університету «Дніпровська політехніка» № 350-с від 16.05.2023р.;
Освітня програма спеціальності 121 «Інженерія програмного забезпечення».

1.4. Постановка завдання

Мета, призначення та техніко-економічна сутність завдання:

Метою даного завдання є створення інноваційного інструменту, який дозволить власникам або операторам сільськогосподарських угідь здійснювати ефективний контроль та управління поливною системою з будь-якого місця.
Основне призначення проекту полягає в поліпшенні ефективності та точності поливної системи за допомогою веб-інтерфейсу. Цей інтерфейс дозволить користувачам з легкістю налаштовувати параметри поливу, відстежувати рівень вологості ґрунту, контролювати роботу системи та отримувати повідомлення про стан системи через Інтернет.

Техніко-економічна сутність завдання полягає в поєднанні технологічних можливостей (віддалений моніторинг та управління) з економічною вигодою (оптимізація ресурсів, зниження витрат). Впровадження веб-інтерфейсу дозволить підвищити ефективність і економічну доцільність використання системи крапельного поливу.

Впровадження веб-інтерфейсу віддаленого моніторингу системи крапельного поливу дозволить оптимізувати використання води, енергії та добрив, що призведе до зменшення затрат на ці ресурси., а також знизити витрати на фізичний контроль та ручне управління поливною системою, оскільки більшість операцій можна виконати з веб-інтерфейсу.

Об'єкти використання розробленої системи:

Розроблений веб-інтерфейс може бути широким, охоплюючи різноманітні об'єкти, де використовуються системи крапельного поливу, такі як сільськогосподарські поля, оранжереї, сади, виноградники, газони, підприємства з вирощування рослин тощо.

Опис структури об'єктів інформаційної системи та перелік показників: Структура об'єктів інформаційної системи включатиме такі компоненти:
база даних для зберігання інформації про стан поливної системи налаштування, історію поливу, датчикові дані; модуль збору даних; веб-інтерфейс для користувачів, що дозволить отримувати доступ до системи, переглядати дані про стан поливної системи.

Показники, що характеризують стан інформаційної системи, можуть включати стан вологості ґрунту, температуру, витрати води, час поливу, історію поливу.

Опис призначення вихідної інформації:

Вихідна інформація, яка буде збиратись та оброблятись системою, включатиме дані про студентів, викладачів, групи, розклад занять, академічні досягнення, студентські документи тощо. Ця інформація буде використовуватись для різних операцій, таких як формування розкладу занять, ведення електронних журналів, видача документів студентам та інше.
Вимоги до організації збору та передачі в обробку вхідної інформації, до контроля і коригування:

Система повинна бути здатною збирати дані з датчиків, які моніторять стан поливної системи, такі як вологості ґрунту, температура, витрата води тощо. Вимоги до збору даних можуть включати надійність, ефективність, безпеку та синхронізацію даних між різними компонентами системи та їх вчасну оновленість.

Умови, при яких припиняється розв'язання завдання автоматизованим способом:

Розв'язання завдання автоматизованим способом може припинятись у разі

відсутності зв’язку або технічних проблем, таких як несправність датчиків, обладнання або програмного забезпечення. Або коли користувач може мати можливість втручатися в роботу системи і змінювати параметри поливу вручну. У такому випадку, якщо користувач здійснює власні налаштування або вимикає автоматичний режим, автоматизований спосіб розв'язання завдання може бути припинений.

Зв'язок даної системи з іншими задачами:

Дана система може мати зв'язок з різними задачами і системами, наприклад, з системою сільськогосподарського управління, системою метеорологічного моніторингу, системою енергетичного управління. Зв'язок з іншими задачами та системами допомагає створити інтегровану та комплексну систему управління крапельним поливом, забезпечуючи зручність, ефективність та оптимальність процесу поливу рослин.

Розподіл функцій між персоналом і технічними засобами при різних ситуаціях вирішення завдань системи:

У різних ситуаціях вирішення завдань системи веб-інтерфейсу віддаленого моніторингу системи крапельного поливу може бути розподілена функціональність між персоналом і технічними засобами. Наприклад, конфігурація та налаштування системи, то в цій ситуації персонал відповідає за конфігурацію та налаштування системи крапельного поливу через веб-інтерфейс.

Вони встановлюють параметри поливу, розклади поливальних циклів, встановлюють межі чутливості датчиків, налаштовують сповіщення та інші параметри, що впливають на роботу системи.

Технічні засоби виконуватимуть завдання збору, обробки та зберігання інформації, а також забезпечуватимуть комунікацію та взаємодію між користувачами системи.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Система повинна надати можливість реєстрації нових користувачів та авторизації вже зареєстрованих користувачів для доступу до системи.
Веб-інтерфейс забезпечувати відображення в режимі реального часу даних про стан системи крапельного поливу, таких як рівень води, тиск, температура та інші важливі показники.

Система може передавати користувачам сповіщення та повідомлення про різні події, такі як низький рівень води, відсутність зв'язку зі станцією поливу або несправності в системі.

1.5.2. Вимоги до інформаційної безпеки

Система повинна мати механізм автентифікації користувачів для перевірки їхньої ідентичності, а також механізм авторизації для контролю доступу до різних функціональних можливостей системи.

Система повинна мати механізми регулярного резервного копіювання даних, а також можливість їх відновлення в разі втрати або пошкодження. Це допоможе забезпечити доступність даних та уникнути втрати важливої інформації.

1.5.3. Вимоги до складу та параметрів технічних засобів

Система передбачає використання клієнтських пристроїв. Це можуть бути смартфони, планшети або комп'ютери, і вимоги можуть стосуватися операційної системи, мінімальних ресурсів, сумісності з веб-браузерами та інших параметрів.

1.5.4. Вимоги до інформаційної та програмної сумісності

Система повинна бути сумісною з різними джерелами вхідної інформації, такими як датчики крапельного поливу, метеорологічні станції та інші джерела даних. Вона повинна забезпечувати збір, обробку та інтеграцію цих даних для подальшого аналізу та прийняття рішень.

Веб-додаток повинен забезпечувати сумісність з різними форматами даних, які можуть використовуватися в аграрному секторі, такими як CSV, JSON, XML тощо. Це дозволить ефективний обмін даними з іншими системами та програмними продуктами.

Система повинна бути розроблена на платформі, яка є поширеною та має велику базу розробників. Це забезпечить доступність технічної підтримки та можливість майбутнього розширення та покращення системи.

Вимоги до апаратного забезпечення повинні бути заздалегідь визначені, щоб система могла ефективно працювати на різних пристроях з різними характеристиками, такими як процесор, оперативна пам'ять, місткість диска тощо.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення програми

Функціональне призначення програми полягає в тому, щоб забезпечити користувачам зручний доступ до інформації про стан системи крапельного поливу, дати їм змогу контролювати та керувати поливом віддалено, а також аналізувати дані для прийняття розумних рішень щодо поливу рослин. Структурну схему, яка пояснює принцип функціонування можна переглянути на рис. 2.1.

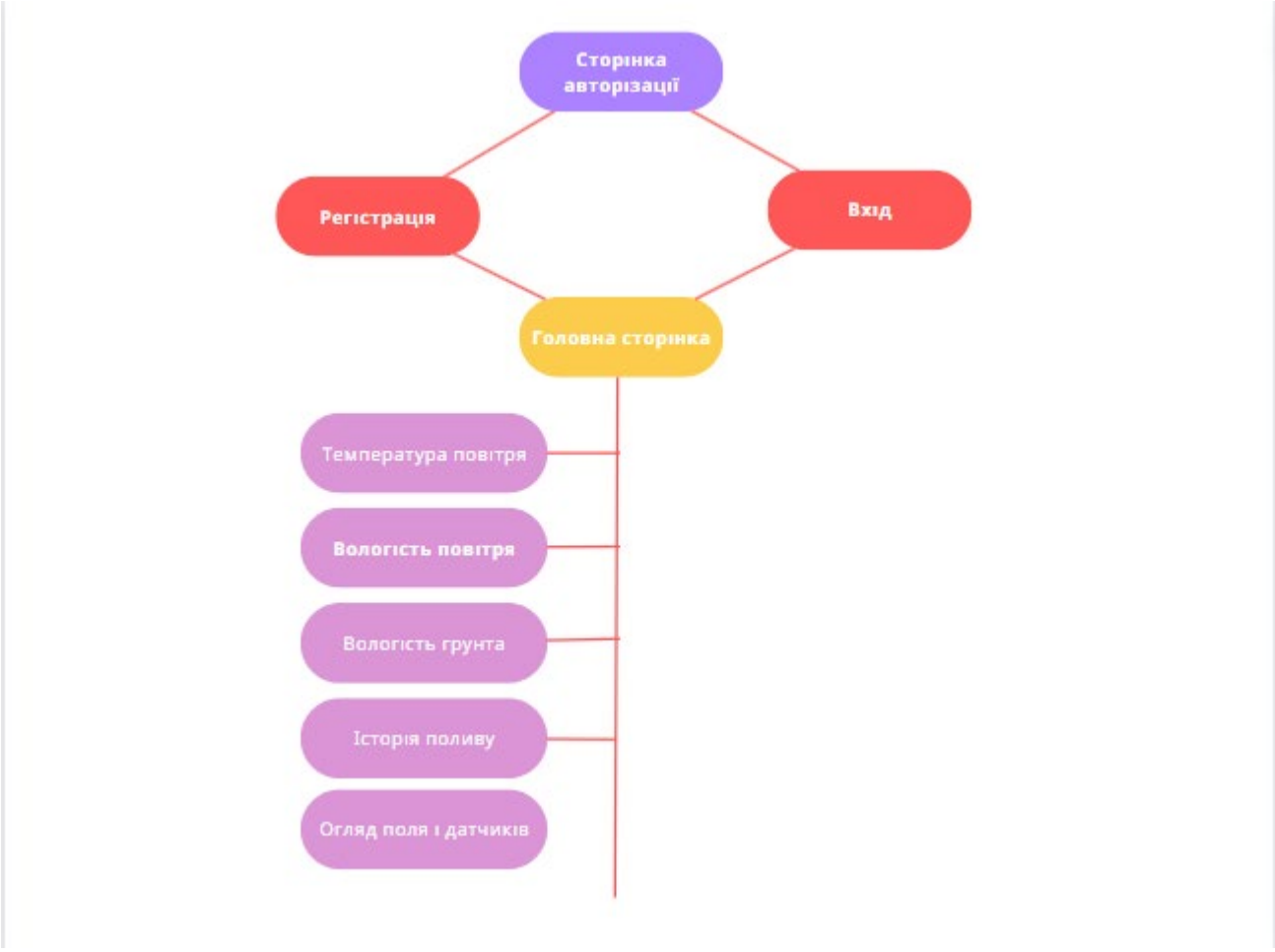

Рис. 2.1. Структурна схема принципу функціонування веб-інтерфейсу

Розроблений веб-інтерфейсу для віддаленого моніторингу та управління системою крапельного поливу має наступні функціональні можливості.

Збір даних: здійснюється збір даних з різних джерел, таких як датчики вологості ґрунту, метеорологічні станції та інші сенсори. Отримується інформація про рівень вологості, температуру, опади та інші фактори, що впливають на полив рослин.

Візуалізація даних: Програма відображає зібрані дані у зручному вигляді, вигляді графіків. Це дозволяє користувачу швидко аналізувати стан поливної системи, виявляти тенденції та здійснювати прийняття рішень щодо поливу.
Віддалене управління: веб-додаток надає можливість віддалено керувати системою крапельного поливу. Користувач може включати або вимикати систему з будь-якого пристрою з доступом до Інтернету.

Експлуатаційне призначення розробленої програми полягає в забезпеченні зручного та ефективного використання системи крапельного поливу з використанням веб-інтерфейсу для віддаленого моніторингу та управління. Основні аспекти експлуатації програми включають:

Моніторинг: Користувачі зможуть отримувати регулярну інформацію про стан системи крапельного поливу, включаючи рівень вологості ґрунту, температуру, опади та інші параметри. Це дозволить їм здійснювати контроль за поливною системою і вчасно реагувати на будь-які проблеми чи незвичайні ситуації.

Управління: Користувачі матимуть можливість віддалено керувати системою крапельного поливу, встановлювати параметри поливу, налаштовувати графіки поливу та контролювати його тривалість та інтенсивність. Це дозволить забезпечити оптимальний полив рослин в залежності від їх потреб та умов оточуючого середовища.

Аналіз та оптимізація: Розроблена програма надасть користувачам зручні інструменти для аналізу зібраних даних про полив, що дозволить їм зрозуміти ефективність системи, виявити тенденції та покращити розподіл ресурсів. Користувачі зможуть аналізувати дані про вологість ґрунту, погоду та інші

фактори для оптимізації поливного режиму та забезпечення збалансованого росту рослин.

2.2. Опис застосованих математичних методів

У розробці додатку не використовувались математичні методи.

2.3. Опис використаної архітектури та шаблонів проектування

Використана архітектура: програма має дві основні компоненти - клієнтську (frontend) і серверну (backend) частини. Клієнтська частина відповідає за інтерфейс користувача, відображення даних та взаємодію з користувачем. Серверна частина забезпечує обробку запитів, зберігання та доступ до даних.
Модель-Контролер (Model-View-Controller, MVC): Шаблон проектування, де програма розділяється на три основні компоненти: модель, контролер та представлення. Модель відповідає за зберігання та обробку даних, контролер - за обробку запитів та керування виконанням, а представлення - за відображення даних користувачу.

Приклад моделі архітектури MVC (Model-View-Controller) представлений на рис. 2.2.

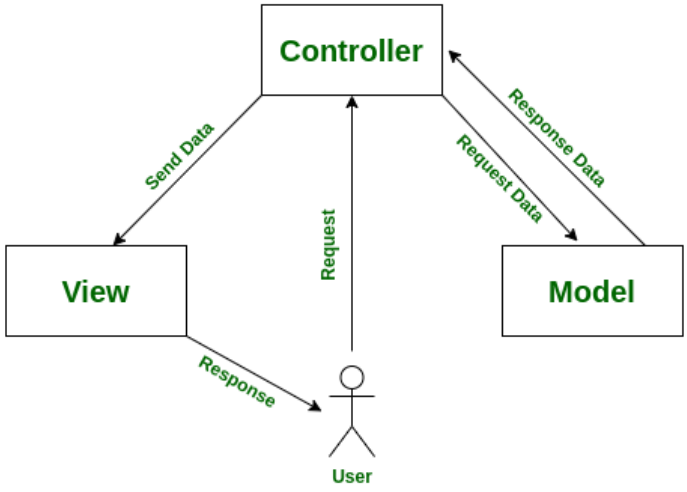

Рис.2.2. Приклад моделі архітектури MVC.

Детальну роботу MVC у розробленому веб-додатку можна переглянути на рис.2.3.

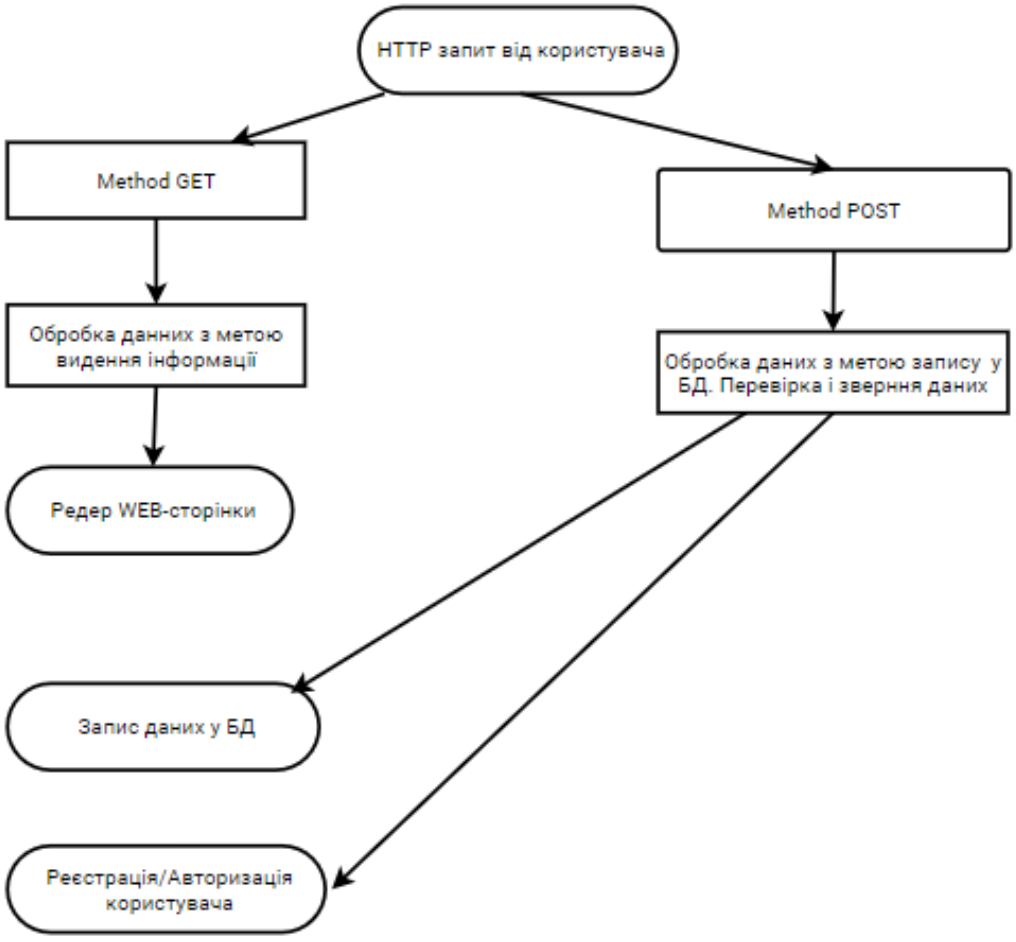

Рис.2.3. Блок схема роботи MVC розробленому веб-додатку

Після отримання HTTP-запиту від користувача, сервер розпізнає його на методи GET і POST. Використання GET-запиту дозволяє користувачеві отримати інформацію, наприклад, перехід на іншу сторінку шляхом зміни URL. У цьому випадку відбувається виведення інформації користувачу без здійснення глобальних операцій з базою даних.

Коли користувач вводить дані за допомогою методу POST, наприклад, заповнює контактну форму, відбувається розширена робота з базою даних, яка включає збереження інформації, відправку данних з

датчиків та здійснення будь-яких глобальних змін в стані програми.

2.4. Опис використаних технологій та мов програмування

HTML (HyperText Markup Language) - це стандартна мова розмітки для створення структури та вигляду веб-сторінок. Вона використовується для описування елементів, їх розташування на сторінці, взаємодії з користувачем та відображення контенту у веб-браузерах.

Основна структура HTML-документа складається з елементів, які заключені в теги. Теги представляються у вигляді пари - відкриваючий тег <tag> та закриваючий тег </tag>. Всі елементи розташовуються в тілі (<body>) документа.

HTML надає різноманітні теги для розмітки різних типів контенту. Наприклад:

<h1>, <h2>, <h3>, ... <h6>: заголовки різних рівнів

<p>: абзац тексту

<a>: гіперпосилання

<img>: зображення

, <ol>, <li>: списки з маркерами або номерами

<table>, <tr>, <td>: таблична структура з рядками та стовпцями

<form>, <input>, <button>: елементи для створення форми та введення даних

, : блоковий та інлайновий контейнери для групування елементів

<header>, <footer>, <nav>, <section>: елементи семантики для розмітки різних частин документа.

HTML також підтримує атрибути, які надають додаткові властивості елементам. Наприклад, class або id для задання ідентифікаторів та класів, src для вказання шляху до зображення, href для встановлення адреси гіперпосилання.

Один з ключових аспектів HTML - це його можливість вкладати елементи один в одного, створюючи ієрархічну структуру. Це дозволяє розташовувати елементи на сторінці в потрібному порядку та створювати складніші макети.

HTML є основною складовою веб-розробки, і використовується спільно з CSS для надання стилів та JavaScript для надання поведінки веб-сторінкам. Використання HTML у вашому проекті дозволить створити структуру сторінок, впорядкувати контент та надати відповідну семантику для кращого розуміння інформації користувачами.

CSS (Cascading Style Sheets) - це мова стилізації, що використовується для надання вигляду та оформлення веб-сторінок, створених з використанням HTML або інших мов розмітки. CSS визначає зовнішній вигляд елементів, таких як кольори, шрифти, розміри, відступи, позиціонування та багато іншого.

Основні концепції CSS включають:

Вибір селекторів: CSS використовує селектори для вибору елементів на сторінці, які потрібно стилізувати. Селектори можуть бути класами, ідентифікаторами, елементами, псевдокласами та багатьма іншими.

Властивості та значення: CSS використовує властивості для визначення конкретних аспектів вигляду елементів. Наприклад, color визначає колір тексту, font-size - розмір шрифту, margin - відступи, background-color - колір фону і т. д. Значення задаються для кожної властивості, наприклад, color: blue або font-size: 16px.

Селектори рівня вкладеності: CSS дозволяє вибирати елементи на основі їх відносного розташування всередині інших елементів. Це дає можливість точно керувати стилями для конкретних елементів.

Каскадність та спадковість: CSS використовує принцип каскаду, що означає, що стилі можуть успадковуватися від батьківських елементів до дочірніх. Це дозволяє ефективно використовувати стилі, задані на вищому рівні, для багатьох елементів.

Використання класів та ідентифікаторів: CSS дозволяє використовувати класи та ідентифікатори для вибору конкретних елементів. Це дозволяє стилізувати окремі елементи або групи елементів з однаковим класом або ідентифікатором.

CSS використовується для створення привабливого та зручного для

використання веб-інтерфейсу. Ваш проект "Розробка веб-інтерфейсу віддаленого моніторингу системи крапельного поливу" може використовувати CSS для надання стильового оформлення, включаючи вибір кольорів, шрифтів, розмірів елементів, позиціонування, відступів та багато іншого. Крім того, CSS може використовуватися для створення адаптивного дизайну, що дозволяє вашому інтерфейсу коректно відображатися на різних пристроях та розмірах екрану.

JavaScript є високорівневою, інтерпретованою мовою програмування, що використовується для розробки веб-додатків. Вона надає можливість динамічно взаємодіяти з елементами сторінки, змінювати їх стан, обробляти події та виконувати різноманітні завдання.

JavaScript має синтаксис, схожий на інші мови програмування, такі як C, C++ та Java. Вона підтримує рядкові, числові, логічні типи даних, а також масиви, об'єкти та функції. За допомогою JavaScript можна виконувати операції з даними, створювати умовні конструкції, цикли та обробляти виняткові ситуації.

JavaScript дозволяє взаємодіяти з елементами HTML-сторінки та їх стилями за допомогою DOM (Document Object Model). За допомогою DOM можна змінювати вміст елементів, створювати нові елементи, змінювати їх атрибути та стилі. Також JavaScript надає можливість обробляти події, такі як клік, наведення курсору миші, введення тексту та багато інших.

Одним із сильних боків JavaScript є підтримка асинхронного програмування. Вона дозволяє виконувати довготривалі операції, такі як відправка запитів на сервер чи завантаження великих обсягів даних, без блокування інтерфейсу користувача. Для цього використовуються функції зворотного виклику (callback functions), обіцянки (promises) та асинхронні функції (async/await).

JavaScript також може працювати на боку сервера за допомогою платформи Node.js[7-10]. Використання Node.js дозволяє створювати серверні додатки, оброблювати запити, взаємодіяти з базами даних та забезпечувати

зв'язок між клієнтом та сервером.

У моєму проекті JavaScript використовується для розробки веб-інтерфейсу віддаленого моніторингу системи крапельного поливу. Де можна взаємодіяти з елементами сторінки, оновлювати дані в реальному часі, обробляти події, відправляти та отримувати дані з сервера. JavaScript дозволяє створити динамічний інтерфейс, який забезпечить зручну та ефективну роботу з системою моніторингу крапельного поливу.

2.5. Опис структури системи та алгоритмів її функціонування

Node.js - це відкрите середовище виконання JavaScript, побудоване на рушії Chrome V8. Воно дозволяє виконувати JavaScript на стороні сервера і розробляти швидкі та ефективні веб-додатки.

Основні характеристики Node.js:

Асинхронність: Node.js працює за асинхронною моделлю вводу/виводу (I/O). Це означає, що він може обробляти багатооперативні запити без блокування виконання інших операцій. Це робить його дуже ефективним для розробки серверних додатків, які потребують обробки багатьох запитів одночасно.

Модульність: Node.js підтримує модульну архітектуру, що дозволяє розділяти функціональність на невеликі модулі. Кожен модуль має свою внутрішню логіку, і його можна легко включити або виключити з проекту. Це полегшує розробку, тестування та підтримку коду.

Пакетний менеджер npm: Node.js поставляється з великим репозиторієм пакетів npm, який містить тисячі використовуваних сторонніх пакетів. За допомогою npm можна легко установлювати, оновлювати та керувати залежностями проекту. Це значно спрощує розробку, оскільки ви можете використовувати готові рішення замість написання коду з нуля.

Підтримка сетевих протоколів: Node.js має вбудовану підтримку різних сетевих протоколів, таких як HTTP, HTTPS, TCP і UDP. Це дозволяє

розробникам створювати різноманітні мережеві додатки, такі як веб-сервери, чат-додатки, API і багато інших.

Швидкість та продуктивність: Node.js базується на руші V8, яке відповідає за виконання JavaScript-коду. V8 від Google є одним з найшвидших рушіїв JavaScript, що забезпечує високу продуктивність Node.js додатків.

Розширюваність: Node.js підтримує використання різних модулів, що дозволяє розширювати його функціональність. Ви можете використовувати сторонні модулі або створювати власні модулі для розширення можливостей Node.js за вашими потребами.

Node.js є популярним вибором для розробки веб-інтерфейсів та серверних додатків. Його асинхронність, модульність і широкі можливості дозволяють легко розробляти швидкі та масштабовані додатки, які відповідають вимогам розробки веб-інтерфейсу віддаленого моніторингу системи крапельного поливу.

NPM модулі які використовуються в проекті:

– "axios": "^1.4.0",
– "body-parser": "^1.20.2",
– "chart.js": "^4.3.0",
– "ejs": "^3.1.9",
– "express": "^4.18.2",
– "express-session": "^1.17.2",
– "fs": "^0.0.1-security",
– "mongodb": "^5.6.0",
– "mongoose": "^6.11.2",
– "passport": "^0.5.2",
– "passport-local": "^1.0.0",
– "passport-local-mongoose": "^6.3.0"

Express[11] є популярним фреймворком веб-додатків для Node.js. Він надає простий та ефективний спосіб створення веб-додатків та API. Особливості Express включають:

Маршрутизація: Express дозволяє визначати маршрути та оброблювати різні HTTP-запити (GET, POST, PUT, DELETE і т.д.) до цих маршрутів. Маршрутизація дозволяє організувати структуру додатку та логічно розділити різні функціональні частини.

Обробка middleware: Middleware - це функції, які обробляють запити перед тим, як вони досягнуть своєї обробки. Express дозволяє додавати middleware для виконання різних завдань, таких як аутентифікація, перевірка даних, реєстрація журналів тощо.

Шаблонізація: Express має вбудовану підтримку шаблонів, що дозволяє відокремлювати представлення від бізнес-логіки. Це дозволяє розробникам створювати динамічні HTML-сторінки, використовуючи шаблони.

Обробка статичного контенту: Express дозволяє просто обробляти статичний контент, такий як зображення, CSS-файли, JavaScript-файли тощо. Він надає можливість визначити папку зі статичним контентом і автоматично обслуговувати файли з цієї папки.

Розширені можливості: Express дозволяє розширити функціональність додатку за допомогою різних пакетів middleware та доповнень. Це дає можливість додатково налаштовувати та розширювати функціональність додатку за потреби.

Express є популярним вибором для розробки веб-додатків на Node.js завдяки своїй простоті, гнучкості та активному співтовариству розробників. Він дозволяє ефективно реалізовувати різноманітні сценарії веб-розробки та розширювати можливості свого проекту.

Саме база даних MongoDB використовується у моїй програмі, де зберігаються данні про користувачів, та дані з датчиків для подальшого їх відображення.

MongoDB - це документ-орієнтована NoSQL база даних, яка зберігає дані у вигляді документів у форматі JSON (JavaScript Object Notation). Вона використовується для зберігання та управління даними веб-додатків, в тому числі для проектів, пов'язаних з веб-інтерфейсом віддаленого моніторингу

системи крапельного поливу.

MongoDB пропонує декілька ключових можливостей, які роблять її привабливим вибором для такого проекту:

Гнучкість: MongoDB дозволяє зберігати структуровані дані без вимог до заздалегідь визначеної схеми. Це особливо корисно, коли розробка системи крапельного поливу може залежати від різних факторів, які можуть змінюватися з часом. MongoDB дозволяє додавати нові поля до документів без необхідності зміни структури бази даних.

Швидкодія: MongoDB пропонує широкий спектр оптимізацій для досягнення високої продуктивності та швидкодії операцій з базою даних. Вона підтримує індекси для прискорення пошуку та запитів, а також може використовувати розподілені системи для розширення масштабування.

Горизонтальне масштабування: MongoDB може працювати в розподіленому середовищі, дозволяючи горизонтально масштабувати базу даних. Це означає, що ви можете додавати нові сервери для збільшення обсягу даних та підвищення продуктивності.

Запити і агрегація: MongoDB має потужний механізм запитів і агрегації, який дозволяє виконувати різні операції над даними, включаючи фільтрацію, сортування, групування та обчислення агрегатних функцій. Це дозволяє легко отримувати необхідну інформацію з бази даних для моніторингу системи.

MongoDB є потужним інструментом для зберігання та управління даними у веб-проектах. Його гнучкість, швидкодія і можливості масштабування роблять його відмінним вибором для проекту з розробкою веб-інтерфейсу віддаленого моніторингу системи крапельного поливу.

Структурну схему, яка ілюструє зв’язок описаних компонентів можна побачити на рис. 2.4.

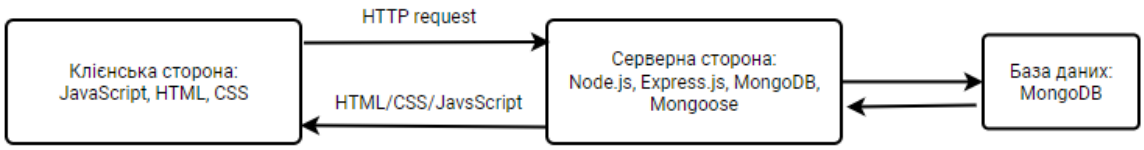

Рис.2.4. Структурна схема взаємодії компонентів

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Організація вхідних та вихідних даних в програмі для розробки веб-інтерфейсу віддаленого моніторингу системи крапельного поливу є ключовим етапом проектування та розробки інформаційної системи.

2.6.1. Характер, організація і попередня підготовка вхідних даних

Сенсорні дані збираються з датчиків, які моніторять різні параметри системи поливу, такі як вологість ґрунту, температура, освітленість та інші. Ці дані представляються у вигляді числових значень, що оновлюються в реальному часі.

Організація вхідних даних включає збір інформації з датчиків за допомогою відповідних пристроїв або модулів, які підключені до системи поливу. Ці дані можуть передаватися через бездротові або провідні з'єднання до центральної системи, де вони оброблюються та зберігаються для подальшого використання.

2.6.2. Характер і організація вихідних даних

Вихідні дані в моїй роботі це перш за все візуалізація даних, що є основною задачею програми - надати користувачам зрозумілу та зручну візуалізацію даних про стан системи поливу. Це може включати відображення графіків, діаграм,

карт, таблиць або інших елементів, які показують різні параметри поливу, такі як вологість ґрунту, температура, рівень води та інші фактори. Формат, опис і спосіб кодування вхідних та вихідних даних. Вхідні об’єкти представлені в таблиці 2.1.

Таблиця 2.1

Вхідні об’єкти

Метод	Дані	Опис
addUser	id	Id користувача
	First_name	Ім’я
	Last_name	Прізвище
	email	Логін користувача
	password	пароль

2.7. Опис розробленої програми

2.7.1 Використані технічні засоби

Враховуючи потреби сучасного інформаційного середовища та вимоги до функціональності додатку, було здійснено вибір найкращих технічних рішень, що забезпечують ефективність та надійність системи.

Для реалізації десктопного додатку були використані наступні технічні засоби:

Комп'ютерна система:

Процесор: Інтел Core i5-4200M з тактовою частотою 2.5 ГГц та 2 ядрами, щозабезпечує потужні обчислювальні можливості та швидку обробку даних.

Оперативна пам'ять: 8 ГБ DDR4 з тактовою частотою 3200 МГц, що дозволяє забезпечити швидкий доступ до даних та підтримувати багатозадачність.

Жорсткий диск: SSD ємністю 240 ГБ, що забезпечує швидкий доступ до

файлів та даних додатку.

Відеокарта: Intel HD, що дозволяє забезпечити завовільну продуктивність графічного інтерфейсу додатку.

Операційна система: Microsoft Windows 10: стабільна та популярна операційна система, яказабезпечує широкі можливості розробки та сумісність з багатьма програмними засобами.

Цей склад технічних засобів був обрано з метою забезпечення швидкої та ефективної роботи десктопного додатку для деканату.

Об'єднуючи обчислювальні можливості, стабільність операційної системи, зручність розробки та зберігання даних, ці технічні засоби дозволяють забезпечити високу продуктивність та функціональність системи управління деканатом.

2.7.2. Використані програмні засоби

Для зручної розробки програмного забезпечення використовуються спеціалізовані програми, які надають багато функціональних можливостей і полегшують процес розробки. Незважаючи на те, що будь-який текстовий редактор, такий як текстовий блокнот або Microsoft Word, може використовуватись для написання програмного коду, професійні розробники використовують спеціалізовані редактори, які надають більше можливостей та підтримку при розробці програмного забезпечення.

Наприклад, деякі популярні редактори програмного коду включають:

Sublime Text 3: Це популярний редактор з великою кількістю розширень та можливостей, таких як підсвічування синтаксису, автодоповнення, швидкий перехід по файлам і багато іншого.

Visual Studio Code[12]: Це безкоштовний та потужний редактор, розроблений компанією Microsoft. Він має широкий набір функцій, включаючи вбудовану підтримку Git, розширення для різних мов програмування,

налагоджувач та інтегровану консоль.

Atom: Це розширюваний редактор з відкритим вихідним кодом, розроблений командою GitHub. Він пропонує багато функцій, таких як автодоповнення, керування пакетами, розмітка коду та інші налаштування.
WebStorm: Це інтегроване середовище розробки (IDE), розроблене компанією JetBrains. Воно спеціалізується на розробці веб-додатків і надає розширену підтримку для JavaScript, HTML, CSS та інших веб-технологій.

Ці редактори надають зручне середовище для написання, редагування та налагодження програмного коду, забезпечуючи розробникам більшу продуктивність та ефективність при роботі з проектами програмного забезпечення.

Для розробки свого веб-сайту я обрав текстовий редактор Visual Studio Code Visual Studio Code (VS Code) - це легкий, швидкий і потужний редактор коду, розроблений компанією Microsoft. Він надає багато функцій, які роблять розробку програмного забезпечення зручнішою і продуктивнішою.

Однією з основних переваг VS Code є його розширюваність. Він має велику кількість розширень, які дозволяють розширити його функціональність для різних мов програмування і технологій. Розширення дозволяють використовувати додаткові функції, такі як автодоповнення коду, інтеграцію з системами контролю версій, налагоджувальник і багато іншого. Це дозволяє програмістам налаштовувати редактор під свої потреби і працювати зручніше.
Ще однією перевагою VS Code є його інтуїтивний і простий у використанні інтерфейс. Він має зрозумілу структуру та інтуїтивний набір інструментів, що полегшує навігацію та редагування коду. Також, він підтримує багато корисних функцій, таких як швидке переключення між файлами, пошук та заміна, рефакторинг коду та інше.

Ще однією суттєвою перевагою VS Code є його багатий екосистема та підтримка спільноти розробників. Він має активну спільноту користувачів, яка створює нові розширення, публікує корисні плагіни та надає підтримку один одному. Крім того, Microsoft постійно вдосконалює VS Code, випускаючи

оновлення та виправляючи помилки, що забезпечує стабільну та актуальну роботу з редактором.

Загалом, Visual Studio Code є потужним і зручним редактором коду, який варто вибрати для розробки програмного забезпечення. Його розширюваність, простий інтерфейс та підтримка спільноти розробників роблять його популярним серед програмістів. Вигляд робочого вікна програми можна побачити на рис. 2.5.

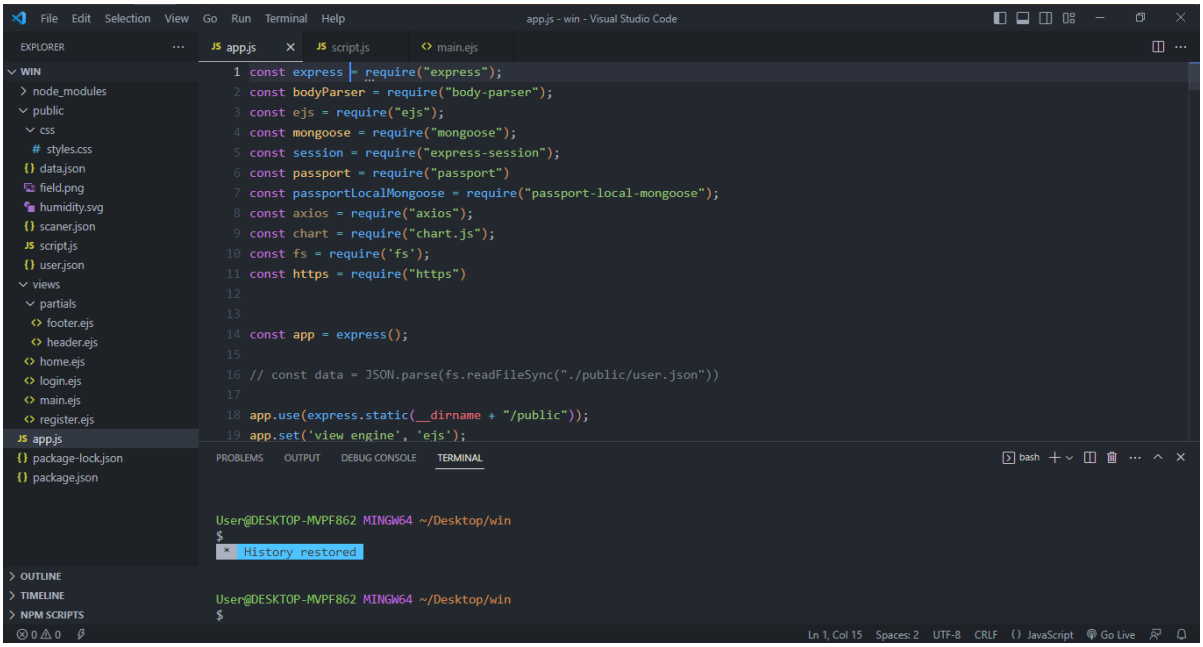

Рис. 2.5. Вигляд робочого вікна програми Visual Studio Code

Для комфортної роботи з базою даних MongoDB найбільш підходить програма MongoDB Compass.

MongoDB Compass - це графічний інтерфейс користувача (GUI) для роботи з базою даних MongoDB. Він надає зручну та інтуїтивно зрозумілу візуалізацію даних та можливості управління базою даних.

Однією з головних переваг MongoDB Compass є його здатність візуалізувати структуру та зміст документів у колекціях бази даних. Він надає зручний графічний інтерфейс для перегляду, фільтрації та редагування документів, що значно полегшує розуміння та редагування даних. Вигляд робочого вікна програми можна побачити на рис. 2.6.

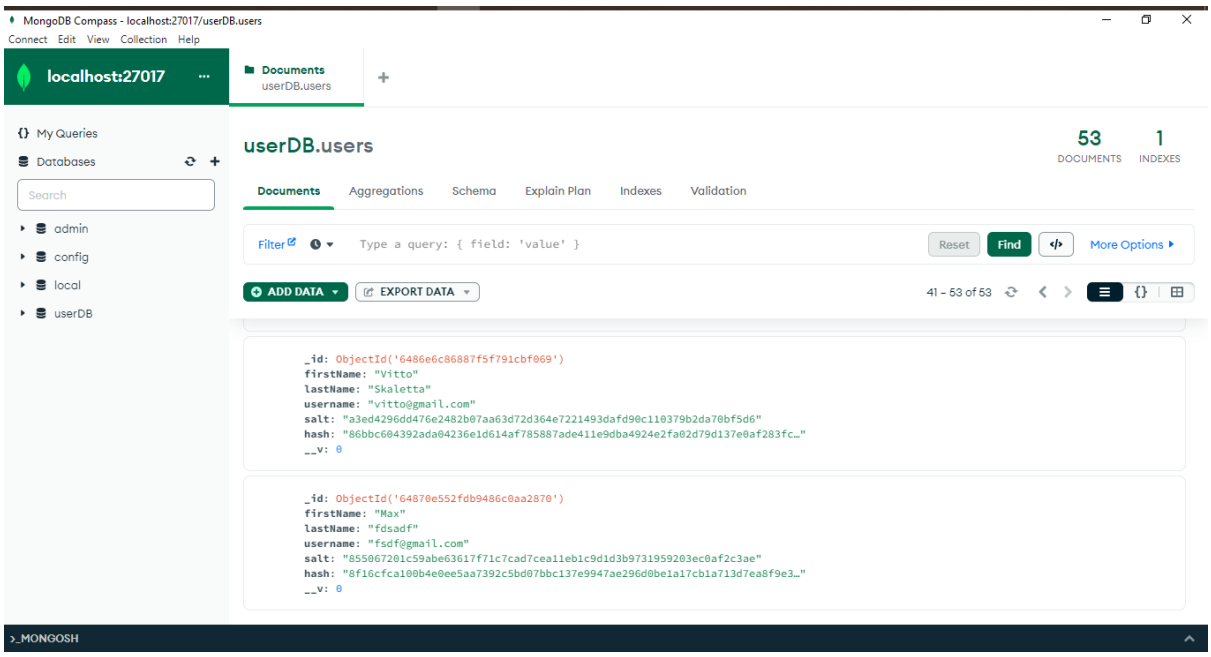

Рис. 2.6. Вигляд робочого вікна програми MongoDB Compass

2.7.3 Виклик та завантаження програми

Для запуску веб-додатків, такого як веб-інтерфейс для віддаленого моніторингу системи крапельного поливу, можна використовувати стандартні методи запуску програм з відповідного носія даних. Програма може бути завантажена на різні пристрої, такі як ПК, ноутбуки, планшети або смартфони з різними типами та конфігураціями, під управлінням різних операційних систем. Для доступу до веб-сайту не потрібно встановлювати додаткове програмне забезпечення. Лише потрібно знати доменне ім'я вашого веб-сайту і зробити запит до нього за допомогою веб-браузера. Веб-браузер виконує роль клієнта, який взаємодіє з веб-сайтом, відображає сторінки та дозволяє вам користуватись функціональністю вашої програми.

Отже, для запуску програми необхідний веб-браузер, який є стандартним для багатьох пристроїв, і встановлення програми не є обов'язковим. Можна отримати доступ до веб-інтерфейсу, просто завантаживши сторінку веб-сайту у своєму веб-браузері.

2.7.4 Опис інтерфейсу користувача

Перш ніж ви зможете вважатися користувачем системи, ви повинні створити обліковий запис або пройти автентифікацію. На рис. 2.7 нижче показано скріншот сторінки реєстрації.

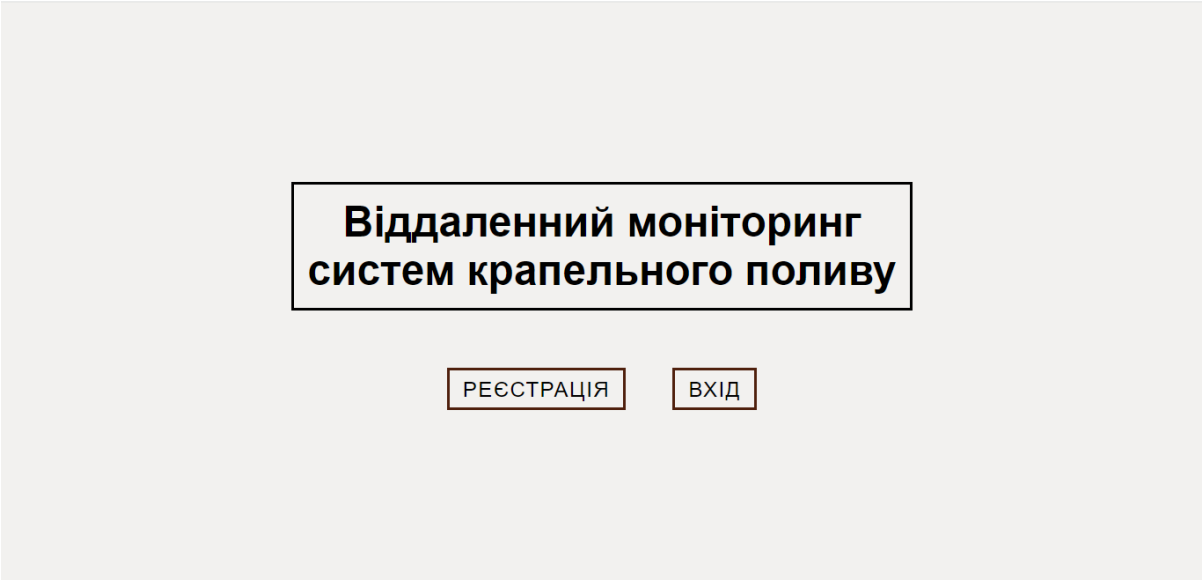

Рис. 2.7. Стартова сторінка веб-додатка

Якщо користувач натиснув на кнопку під назвою «Реєстрація», то перед ним відкриється наступна сторінка з формою реєстрації (рис 2.8).

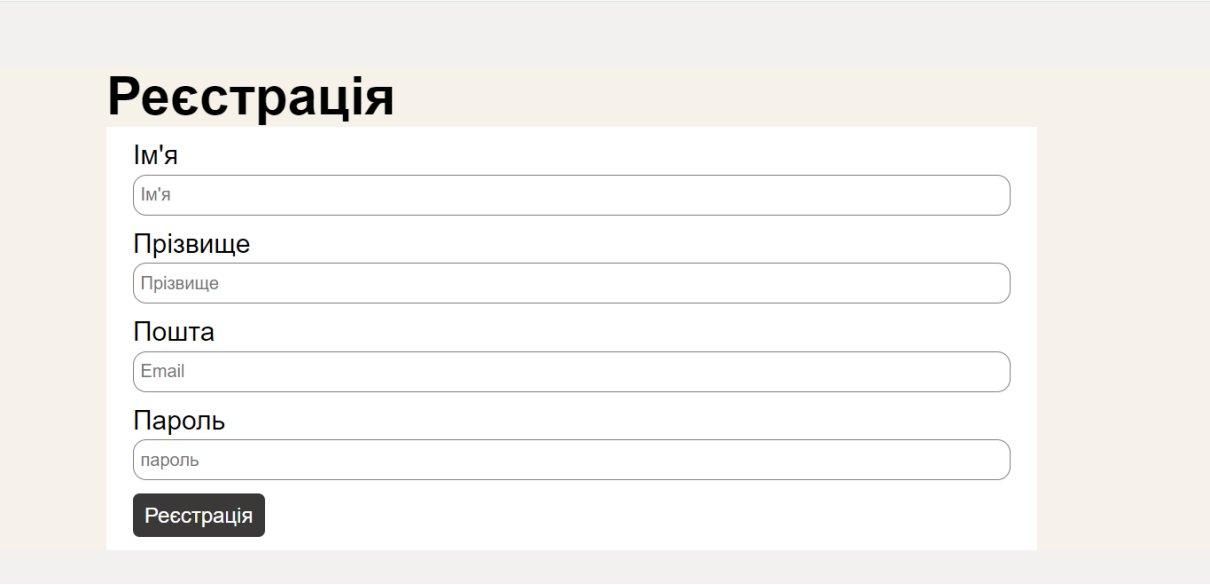

Рис. 2.8. Форма реєстрації

Після реєстрації, або якщо спочатку натиснути кнопку «Вхід». Потрібно увійти в систему, вказавши адресу електронної пошти та пароль, як показано на рис. 2.9.

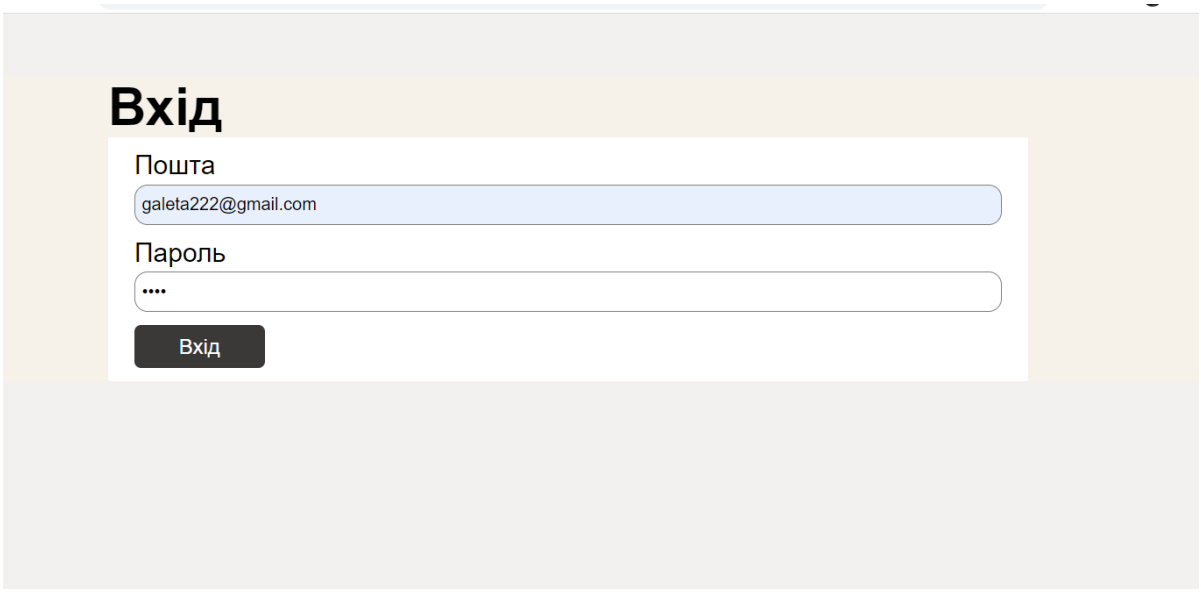

Рис. 2.9. Сторінка входу в систему

Після автентифікації ми потрапляємо на головну сторінку (рис. 2.10).

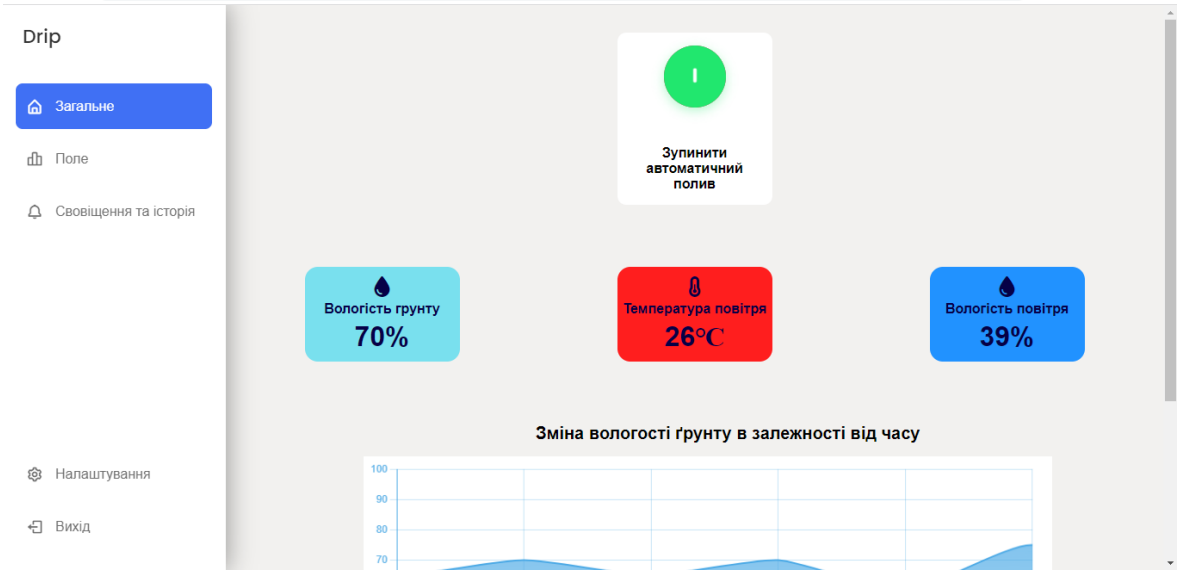

Рис. 2.10. Головна сторінка (частина 1)

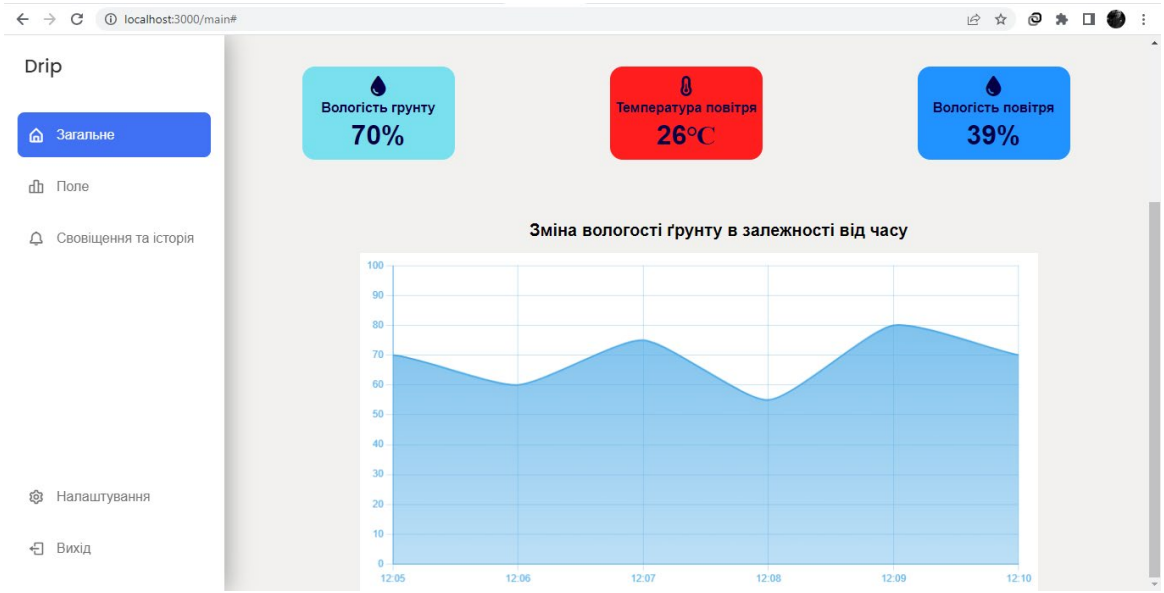

Рис. 2.11. Головна сторінка (частина 2)

З домашньої сторінки можна отримати доступ до ряду функцій, включаючи всі ті, що перераховані на лівій бічній панелі.

На головній сторінкі ми бачимо чотири квадрати, які відображають відповідну інформацію з першого погляду. Також видно один графік, який відображає зміну вологості ґрунту в залежності від часу, і кожні 10 хв в режимі реального часу оновлюється, коли надходять нові данні з датчиків.

Натистувши в лівій панелі вкладку «Поле», користувач зможе побачити

своє поле і датчики (рис. 2.11). Де самі датчики, кожні 10 хв обновлюються і показують рівень вологості ґрунту. І в залежності від відсотка вологи, змінюють свої кольори. Всього є три кольори, які можуть прийняти датчики.

Також для прикладу обрана культура соняшник, для якої оптимальна вологість кореневого шару для соняшника становить 60-70%.

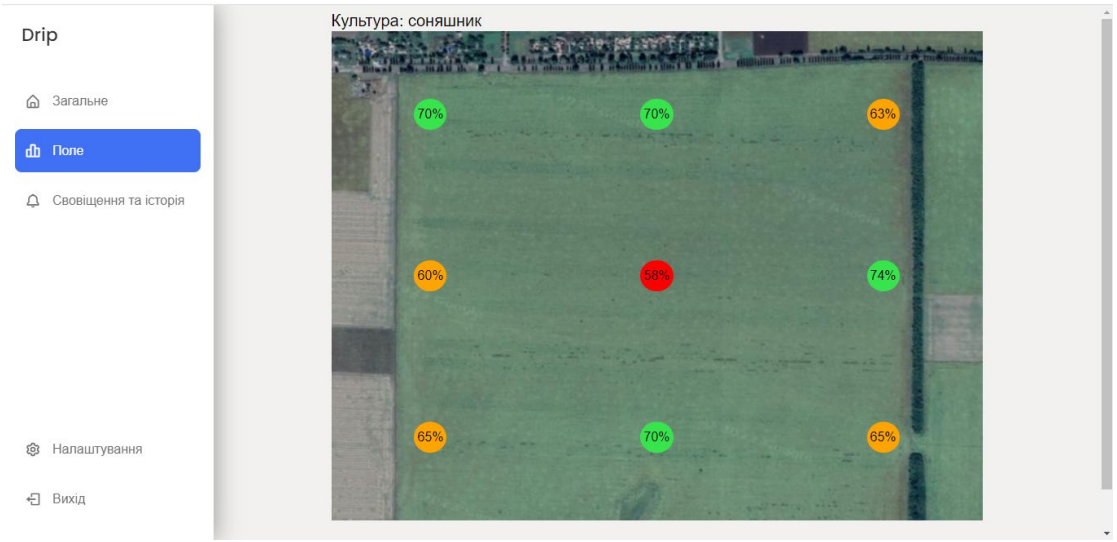

Рис. 2.11. Сторінка поля з датчиками

Ця сторінка (рис. 2.12), дає нам доступ до історії поливу з усіма необхідними деталями.

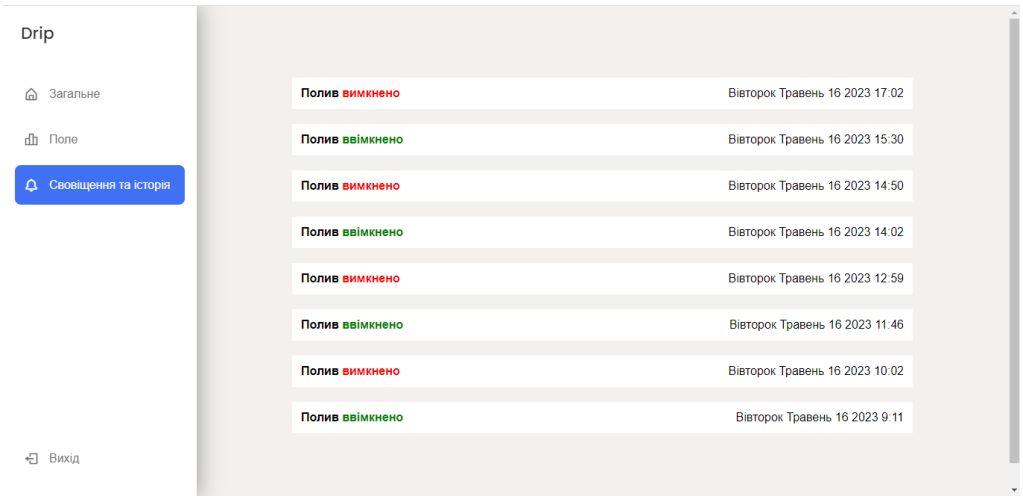

Рис. 2.12. Сторінка історії поливу

Завдяки історії поливу користувач може аналізувати та контролювати ефективність системи, визначати оптимальний режим поливу, виявляти проблеми та здійснювати налагодження. В результаті, історія поливу сприяє покращенню ефективності та оптимізації системи крапельного поливу

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: 1587;

2) Коефіцієнт складності програми: 1,4;

3) Коефіцієнт кореляції програми під час розробки: 0,3;

4) Середня годинна заробітна плата програміста: 115 грн/год;

5) Вартість однієї машино-години ЕОМ: 10 грн/год;

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 веб продукту можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (4.1)$$

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

$t_и$ - витрати праці на дослідження алгоритму рішення задачі;t_a- витрати праці на розробку блок-схеми алгоритму;

$t_п$-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 упрограмногму забезпеченні, яке розробляється.

Умовне число операторів:

$$Q = q \cdot C \cdot (1 + p), \quad (4.2)$$

q - передбачуване число операторів;

С - коефіцієнт складності програми;

р - коефіцієнт корекції програми в ході її розробки; Звідси умовне число операторів в програмі:

$$Q = 1587 \cdot 1{,}4 \cdot (1 + 0{,}3) = 2888$$

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

$$t_u = (Q \cdot B)/(75 \ldots 85)K, \text{ людино-годин}, \quad (3.3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього описузадачі, В=1.4;

K – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи зданої спеціальності: 1.35;

$$t_u = (2888 \cdot 1{,}35) / (80 \cdot 1{,}3) = 63{,}35 \text{ , людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = Q /(20…25)K,\ \text{людино-годин.} \qquad (3.4)$$

$$t_a = 2888 / (20 \cdot 1{,}3) = 187{,}72\ ,\ \text{людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = Q/(20…25)K,\ \text{людино-годин.} \qquad (3.5)$$

$$t_n = 2888 / (22 \cdot 1{,}3) = 170{,}65\ ,\ \text{людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = Q/(4…5)K,\ \text{людино-годин,} \qquad (3.6)$$

$$t_{отл} = 2888 / (5 \cdot 1{,}3) = 751,\ \text{людино-годин,}$$

- за умови комплексного налагодження завдання:

$$t^k_{отл} = 1.5 \cdot t_{отл},\ \text{людино-годин} \qquad (3.7)$$

$$t^k_{от} = 1{,}2 \cdot 751 = 9012,\ \text{людино-годин}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до} \qquad (3.8)$$

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

$$t_{др} = Q /(15...20)K \quad (3.9)$$

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

$$t_{до} = 0.75 \cdot t_{др} \quad (3.1)$$

Підставляємо дані у формулу і отримуємо:

$t_{др} = 2888 / (17 \cdot 1,3) = 171,8$, людино-годин.

$t_{до} = 0,75 \cdot 171,18 = 128,4$, людино-годин.

$t_{д} = 171,18 + 128,4 = 299,58$ людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:t =

50+63,35+18,72+170,65+901+299,58=1672,3 людино-годин.

У результаті ми розрахували, що в загальній складності необхідно1672,3 людино-годин для розробки даного програмного забезпечення.

3.2 Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн} \quad (3.10)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн} \qquad (3.11)$$

де: t - загальна трудомісткість, людино-годин;

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

З урахуванням того, що середня годинна зарплата програміста становить177 грн / год, отримуємо:

$$З_{ЗП} = 1673{,}3 \cdot 115 = 192\ 314{,}5 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми наЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot С_{мч}, \text{ грн}, \qquad (3.12)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ,

год;$С_{мч}$ - вартість машино-години ЕОМ, грн/год.

Підставивши в формулу (3.3) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 901 \cdot 0{,}38 = 342 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 192\ 314{,}5 + 342 = 192\ 656{,}5 \text{ грн}$$

Очікуваний період створення ПЗ:

$$T = t /(B_k \cdot F_p)\text{міс.} \quad (3.13)$$

де B_k - число виконавців (дорівнює 1);

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

Очікуваний період створення ПЗ:

$$T = 1672,3/1 \cdot 176 = 9,5 \text{ міс.}$$

Висновки. Час розробки даного програмного забезпечення складає 1673.3 людино-годин. Таким чином, очікувана тривалість розробки складе 9,5 місяця при 40 годинному робочому тижні (місячний фонд робочого часу 176 годин), а витрати на створення програмного забезпечення складатимуть 201324,5 грн.

ВИСНОВКИ

В ході роботи на тему "Розробка веб-інтерфейсу віддаленого моніторингу системи крапельного поливу" була проведена розробка імплементації веб-інтерфейсу, який дозволяє віддалено моніторити та керувати системою крапельного поливу. Цей веб-інтерфейс надає користувачам зручний спосіб контролювати рівень вологості, графіки поливу та інші параметри, пов'язані з системою поливу.

В процесі розробки було використано сучасні технології веб-розробки, включаючи HTML, CSS і JavaScript. Було створено веб-інтерфейс, який має зручну інтерактивну взаємодію з користувачем, забезпечуючи зручний доступ до різних функціональних можливостей системи крапельного поливу.

Основні функції, які були реалізовані у веб-інтерфейсі, включають відображення поточного стану системи поливу, моніторинг рівня вологості у ґрунті, відображення графіків поливу та повідомлення про події пов'язані з системою поливу.

Застосування веб-інтерфейсу віддаленого моніторингу системи крапельного поливу має багато переваг, таких як зручний доступ до інформації про стан системи в будь-який час і з будь-якого місця, оптимальне використання води та енергії, а також забезпечення ефективного поливу рослин.

У результаті роботи було реалізовано веб-інтерфейс віддаленого моніторингу системи крапельного поливу, який забезпечує зручну інтерактивну взаємодію з користувачем та допомагає оптимізувати процес поливу рослин. Одним із можливих покращень може стати розробка функції налаштування розкладу поливу.

В економічному розділі визначено трудомісткість розробленої інформаційної системи 1673,3 люд-год, проведений підрахунок вартості роботи по створенню програми 201 324,5 грн. та розраховано час на його створення 9,5 місяців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Довідник по HTML тегам URL: https://css.in.ua/html/tags (дата звернення 05.05.2023)

2. Довідник по CSS властивостям URL: https://css.in.ua/css/properties (дата звернення 09.05.2023).

3. Чому JavaScript — перспективна мова програмування? URL: https://dou.ua/lenta/articles/how-to-learn-java/ (дата звернення 03.05.2023).

4. MongoDB Documentation URL: https://www.mongodb.com/docs/ (дата звернення 14.05.2023).

5. Connect MongoDB database to Express server — step-by-step URL: https://medium.com/featurepreneur/connect-mongodb-database-to-express-server-step-by-step-53e548bb4967 (дата звернення 15.05.2023).

6. King, A. Techonolgy: The future of agriculture. Nature 2017 URL : https://www.nature.com/articles/544S21a (дата звернення 03.05.2023).

7. Mongoose Documentation URL: https://mongoosejs.com/docs/guide.html (дата звернення 14.05.2023).

8. Node.js для початківців: опис, керівництво, особливості URL: https://hi-news.pp.ua/kompyuteri/10124-nodejs-dlya-pochatkvcv-opis-kervnictvo-osoblivost.html (дата звернення 09.05.2023).

9. Node.js Fetch data from MongoDB Using Mongoose URL: https://www.tutsmake.com/node-js-fetch-data-from-mongodb-using-mongoose/ (дата звернення 21.05.2023).

10. How to install npm packages? URL: https://www.mariokandut.com/how-to-install-npm-packages/ (дата звернення 21.05.2023).

11. Express web framework (Node.js/JavaScript) URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs (дата звернення 10.05.2023).

12. Знайомство з Visual Studio Code URL:

https://romul.name/blog/znayomstvo-z-visual-studio-code/ (дата звернення 15.05.2023).

13. Axios API URL:https://axios-http.com/ru/docs/api_intro (дата звернення 20.05.2023).

14. Що таке JSON. Усе про цей формат передачі даних в інтернеті URL:

https://apix-drive.com/ua/blog/useful/scho-take-jsonl (дата звернення 20.05.2023).

15. Stafford, J.V. Implementing precision agriculture in the 21st century. J. Agric. Eng. Res. URL: https://www.sciencedirect.com/science/article/abs/pii/S0021863400905778?via%3Dihub (дата звернення 20.05.2023).

16. How to retrieve a record from an existing table in a database using JDBC API? URL: https://www.tutorialspoint.com/how-to-retrieve-a-record-from-an-existing-table-in-a-database-using-jdbc-api (дата звернення 21.05.2023).

17. How to create dynamic drop down list in Java from database / URL: https://www.codejava.net/java-ee/jsp/how-to-create-dynamic-drop-down-list-in- jsp-from-database (дата звернення 21.05.2023).

18. Turner, N.C. Measurement of plant water status by the pressure chamber technique. J. Irrig. Sci. 1998 URL: https://link.springer.com/article/10.1007/BF00296704 (дата звернення 21.05.2023).

19. How to install npm packages? URL: https://www.mariokandut.com/how-to-install-npm-packages/ (дата звернення 21.05.2023).

20. Login and Registration Form in Node JS + Express + mongoDB URL: https://www.tutsmake.com/login-and-registration-form-in-node-js-express-mongodb/ (дата звернення 21.05.2023).

ДОДАТОК А

КОД ПРОГРАМИ

```
const express = require("express");
const bodyParser = require("body-parser");
const ejs = require("ejs");
const mongoose = require("mongoose");
const session = require("express-session");
const passport = require("passport")
const passportLocalMongoose = require("passport-local-mongoose");
const axios = require("axios");
const chart = require("chart.js");
const fs = require('fs');
const https = require("https")



const app = express();

// const data = JSON.parse(fs.readFileSync("./public/user.json"))

app.use(express.static(__dirname + "/public"));
app.set('view engine', 'ejs');
app.use(bodyParser.urlencoded({
  extended: true
}));

app.use(session({
  secret: 'Our little secret',
  resave: false,
  saveUninitialized: false
```

```
}));

app.use(passport.initialize());
app.use(passport.session());

mongoose.set('strictQuery', true);
mongoose.connect("mongodb://127.0.0.1:27017/userDB", { useNewUrlParser: true
});



mongoose.connection
  .once("open", () => console.log("Connected"))
  .on("error", error => {
    console.log(error);
  });

const userSchema = new mongoose.Schema({
  firstName: String,
  lastName: String,
  email: String,
  password: String
});

userSchema.plugin(passportLocalMongoose);

const User = new mongoose.model("User", userSchema)



passport.use(User.createStrategy());
```

```
passport.serializeUser(function (user, done) {
  done(null, user);
});

passport.deserializeUser(function (user, done) {
  done(null, user);
});

// console.log(data)
// const sensor = {
//   date: '00:00',
//   value: '500'
// }

// ----------
// const data = JSON.parse(fs.readFileSync('./public/data.json'))

// const saveData = (data, file) => {
//   const finished = (error) => {
//     if (error) {
//       console.log(error)
//       return;
//     }
//   }


//   const jsonData = JSON.stringify(data, null, 2)
//   fs.writeFile(file, jsonData, finished)
//   console.log("saved")
// }
```

```
// setInterval(function () {

//   const newLabel = new Date().toLocaleTimeString().slice(0, 5);
//   const newData = Math.floor(Math.random() * (10 - 1 + 1)) + 1;
//   const person = {
//     name: newLabel,
//     age: newData
//   }

//   saveUser(person)

// }, 600);


// saveUser = (person) => {
//   const newPerson = {
//     name: person.name,
//     age: person.age
//   }

//   data['sensors - ' + person.name ] = newPerson
//   saveData(data, './public/data.json')
// }
// ----------------



// const myData = {
```

```
//   date: "18:00",
//   value: "100"
// }



// const saveDate = (sensor) => {
//   const newDataSensor = {
//     date: sensor.date,
//     value: sensor.value
//   }

//   data[sensor.date] = newDataSensor;
//   console.log(data[sensor.date])
//   saveData(data, "./public/haha.json")

// }
// saveDate(sensor)



// fs.writeFile('./public/user.json', JSON.stringify(data, null, 2), err => {
//   if (err) {
//     console.log(err)
//   } else {
//     console.log("Success")
//   }
// })

app.get("/", (req, res) => {
  res.render("home")
```

```
})

app.get("/login", (req, res) => {
  res.render("login")
})

app.get("/register", (req, res) => {
  res.render("register")
})

app.get("/main", (req, res) => {

  const sensorData = [70,60,65,70,58,70, 63, 74, 65]

  if (req.isAuthenticated()) {
    const url =
"https://api.openweathermap.org/data/2.5/weather?q=Dnipro&appid=b59aad59ab311e
7b5fae6c93cdeb663d&units=metric"
    https.get(url, response =>{
      response.on("data", data =>{
        const weatherData = JSON.parse(data)
        const temp = Math.ceil(weatherData.main.temp);
        const humidity = weatherData.main.humidity;
        const weatherDescription = weatherData.weather[0].description
        console.log(weatherDescription)
        res.render("main", {data1: humidity, data: temp, sensors : sensorData},)
      })
    })

  } else {
```

```
    res.redirect("/login");
  }


});

app.post("/register", (req, res) => {

  const newUser = new User({
    firstName: req.body.userfirsrname,
    lastName: req.body.userlastname,
    username: req.body.username,
  })

  User.register(newUser, req.body.password, (err, user) => {
    if (err) {
      console.log(err)
      res.redirect("/register")
    } else {
      passport.authenticate("local")(req, res, function () {
        res.redirect("/main");
      });
    }
  })
});

app.post("/login", function (req, res) {

  const user = new User({
    firstName: req.body.userfirsrname,
```

```
    lastName: req.body.userlastname,
    username: req.body.username,
    password: req.body.password
  });

  req.login(user, function (err) {
    if (err) {
      console.log(err);
    } else {
      console.log("tyt")
      passport.authenticate("local")(req, res, function () {
        console.log("tyt1")
        res.redirect("/main");
      });
    }
  });

});

app.listen(3000, () => {
  console.log("Server started on port 3000.");
});

@import
url("https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600&dis
play=swap");

* {
  padding: 0;
  margin: 0;
```

```
  box-sizing: border-box;
}

body {
  font-family: 'Josefin+Sans', sans-serif;
  background-color: rgba(242, 241, 239, 255);
}

.autorisation {
  font-size: 20px;
  background-color: #F6F1E9;
  font-family: 'Josefin+Sans', sans-serif;
}

.home__wrapper {
  min-height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction: column;
}

.home__header {
  margin-bottom: 50px;
  text-align: center;
  border: 2px solid black;
  padding: 10px;
}

.home__links {}
```

```
.home__link {
  margin-right: 35px;
  padding: 5px 10px;
  text-decoration: none;
  /* color: #0079FF; */
  color: black;
  background-color: transparent;
  text-transform: uppercase;
  letter-spacing: 1px;
  outline: 2px solid #4F200D;
  transition: .03s all;
}

.home__link:hover {
  color: #F6F1E9;
  background-color: black;
}

.home__link:last-child {
  margin-right: 0;
}

/* Register */


.register__wrapper {
  width: 700px;
  display: flex;
  flex-direction: column;
```

```
  justify-content: center;
  margin: 3rem 0 0 5rem;
}

.register__header {}

.register__form {
  background-color: white;
  padding: 0 20px;
}

.form__wrap {}

.form__group:first-child {
  margin-top: 10px;
}

.form__group {
  display: flex;
  flex-direction: column;
  margin-bottom: 10px;
  gap: 3px;
}

.input__name {}

.input {
  padding: 7px 0 7px 5px;
  border-radius: 10px;
  border: 1px solid gray;
```

```
}

.input:focus {
  outline: none !important;
  border: 1px solid rgb(70, 70, 70);
}

.form__button {
  margin-bottom: 10px;
  padding: 7px;
  width: 15%;
  color: white;
  font-family: 'Josefin+Sans', sans-serif;
  font-size: 16px;
  background-color: rgb(59, 56, 56);
  border: 0;
  border-radius: 5px;
  cursor: pointer;
  transition: .1.5s all;
}

.form__button:hover {
  background-color: white;
  color: rgb(59, 56, 56);
  outline: 1px solid black;

}

/* Login sections */
```

```
.login__wrapper {
  width: 700px;
  display: flex;
  flex-direction: column;
  justify-content: center;
  margin: 3rem 0 0 5rem;
}

.login__form {
  background-color: white;
  padding: 0 20px;
}

.form__group:first-child {
  margin-top: 10px;
}

.form__group {
  display: flex;
  flex-direction: column;
  margin-bottom: 10px;
  gap: 3px;
}

.input__name {}

.input {
  padding: 7px 0 7px 5px;
  border-radius: 10px;
  border: 1px solid gray;
```

```
}

.input:focus {
  outline: none !important;
  border: 1px solid rgb(70, 70, 70);
}

.form__button {
  margin-bottom: 10px;
  padding: 7px;
  width: 15%;
  color: white;
  font-family: 'Josefin+Sans', sans-serif;
  font-size: 16px;
  background-color: rgb(59, 56, 56);
  border: 0;
  border-radius: 5px;
  cursor: pointer;
  transition: .1.5s all;
}

.form__button:hover {
  background-color: white;
  color: rgb(59, 56, 56);
  outline: 1px solid black;
}


/* Main section start */
```

```
.wrapper__main {
  min-height: 100%;
  background: #e3f2fd;
  font-family: "Poppins", sans-serif;
}

nav {
  position: fixed;
  top: 0;
  left: 0;
  height: 70px;
  display: flex;
  align-items: center;
  background: #fff;
  box-shadow: 0 0 1px rgba(0, 0, 0, 0.1);
}

nav .logo {
  display: flex;
  align-items: center;
  margin: 0 24px;
}

.logo .menu-icon {
  color: #333;
  font-size: 24px;
  margin-right: 14px;
  cursor: pointer;
}
```

```
.logo .logo-name {
  color: #333;
  font-size: 22px;
  font-weight: 500;
}

nav .sidebar {
  position: fixed;
  top: 0;
  left: -100%;
  height: 100%;
  width: 260px;
  padding: 20px 0;
  background-color: #fff;
  box-shadow: 0 5px 1px rgba(0, 0, 0, 0.1);
  transition: all 0.4s ease;
  opacity: 1;
  left: 0;
  pointer-events: auto;
  box-shadow: 3px 0px 29px 0px rgba(0, 0, 0, 0.34);
  background-color: white;
}

nav.open .sidebar {
  left: 0;
}

.sidebar .sidebar-content {
  display: flex;
  height: 100%;
```

```
  flex-direction: column;
  justify-content: space-between;
  padding: 30px 16px;
}

.sidebar-content .list {
  list-style: none;
}

.list .nav-link {
  display: flex;
  align-items: center;
  margin: 8px 0;
  padding: 14px 12px;
  border-radius: 8px;
  text-decoration: none;
}

.lists .nav-link:hover {
  background-color: #4070f4;
}

.lists .active {
  background-color: #4070f4;
}

.lists .active .icon,
.lists .active .link {
  color: #fff;
}
```

```
.nav-link .icon {
  margin-right: 14px;
  font-size: 20px;
  color: #707070;
}

.nav-link .link {
  font-size: 16px;
  color: #707070;
  font-weight: 400;
}

.lists .nav-link:hover .icon,
.lists .nav-link:hover .link {
  color: #fff;
}

.main__wrapper {
  min-height: 100vh;
  margin-left: 260px;
}

.canvas__wrapper {
  margin-left: 160px;
}

.name__chart {
  margin-left: 200px;
  margin-bottom: 15px;
  font-size: 20px;
```

```
}

canvas {
  padding: 5px;
  background-color: white !important;

}

p {
  font-size: 16px;
}

.info__date {
  display: flex;
  justify-content: space-around;
  margin: 4.5rem 0;
}

.wrap {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;

  height: 110px;
  width: 180px;
  border-radius: 15px;
  color: #060047;
}
```

```
.air__temperature {
  background-color: #FF1E1E;
}

.air__humidity {
  background-color: #2192FF;
}

.soil__moisture {
  background-color: #79E0EE;
}

.ico {
  font-size: 24px;
}

.soil__header {
  margin: 5px 0;
}

.soil__value {
  font-size: 32px;
  font-weight: 700;
}

.pumb__wrapper {
  height: 200px;
  width: 180px;
  display: flex;
  flex-direction: column;
```

```
  justify-content: space-between;
  align-items: center;
  background-color: white;
  padding: 15px;
  margin: 2rem auto 0 auto;
  border-radius: 10px;
  text-align: center;
}

/* ---------- */

.power {
  display: table;
  cursor: pointer;
}
.power input {
  display: none;
}
.power input + div {
  position: relative;
  overflow: hidden;
  background: #eb2c59;
  border-radius: 50%;
  padding: 18px;
  transition: transform 0.4s ease;
  filter: drop-shadow(0 4px 6px rgba(235, 44, 89, .3));
}
.power input + div:before {
  content: '';
  width: 4px;
```

```
  height: 14px;
  border-radius: 2px;
  background: #fff;
  position: absolute;
  left: 50%;
  z-index: 2;
  margin: 0 0 0 -2px;
  top: 18px;
  opacity: 1;
  transform: scaleY(1) translate(0, 2px);
  transition: opacity 0.4s ease 0.1s, transform 0.9s ease 0.4s;
}
.power input + div:after {
  content: '';
  position: absolute;
  right: 0;
  bottom: 0;
  width: 200%;
  height: 200%;
  border-radius: 50%;
  background: #21e76e;
  transform: translate(100%, 100%);
  transform-origin: right bottom;
  transition: transform 0.8s ease;
}
.power input + div svg {
  width: 36px;
  height: 36px;
  fill: none;
  display: block;
```

```
  stroke: #fff;
  position: relative;
  z-index: 2;
  stroke-width: 5;
  stroke-linecap: round;
  stroke-dasharray: 84;
  stroke-dashoffset: 160;
  transform: translate3d(0, 0, 0);
  transition: stroke-dasharray 0.6s ease 0s, stroke-dashoffset 0.7s ease 0s;
}
.power input + div:active {
  transform: scale(0.96);
}
.power input:checked + div {
  filter: drop-shadow(0 4px 6px rgba(33, 231, 110, .3));
}
.power input:checked + div svg {
  stroke-dasharray: 120;
  stroke-dashoffset: 133;
  transition: stroke-dasharray 0.6s ease 0.45s, stroke-dashoffset 0.7s cubic-
bezier(0.94, -0.1, 0.1, 1.2) 0.2s;
}
.power input:checked + div:before {
  transform: scaleY(0) translate(0, 7px);
  transition: opacity 0.4s ease 0s, transform 0.6s ease 0.3s;
}
.power input:checked + div:after {
  transform: translate(40%, 40%);
}
```

```
/* ---------- */

h2 {
  font-size: 16px;
}



.sensors__date {
  width: 800px;
  height: 600px;
  background: url("../field.png") no-repeat;
  background-size: cover;
}

.sensors__date {
  display: none;
  justify-content: center;
  align-items: center;
  column-gap: 190px;
  margin: auto;
  margin-top: 2rem;
  position: relative;
}

.sensors__date::before {
  position: absolute;
  left: 0;
  top: -1.5rem;
```

```
  content: 'Культура: соняшник';
  font-family: 'Josefin+Sans', sans-serif;
  font-size: 20px;
  color: black;
}

.sensor__date {
  color: white;
  margin: 10rem 1.5rem;
  background-color: #38E54D;
  color: black;
  padding: 10px 4px;
  border-radius: 50%;
  cursor: pointer;
}

.pora {
  background-color: red;
}

.pora1{
  background-color: orange;
}



.history__wrapper {
  display: none;
  padding: 6rem 8rem;
}
```

```
.history__container {
  display: flex;
  justify-content: space-between;
  background-color: white;
  padding: 12px;
  margin-bottom: 20px;
}


/* Main section end */

<%- include('partials/header') %>

  <div class="register autorisation">
    <div class="register__wrapper">
      <h1 class="register__header">Реєстрація</h1>
      <form action="/register" class="register__form" method="POST">
        <div class="form__wrap">
          <div class="form__group">
            <label for="" class="input__name">Ім'я</label>
            <input type="text" class="input" name="userfirsrname" placeholder="Ім'я">
          </div>
          <div class="form__group">
            <label for="" class="input__name">Прізвище</label>
            <input type="text" class="input" name="userlastname"
placeholder="Прізвище">
          </div>
          <div class="form__group">
            <label for="" class="input__name">Пошта</label>
```

```
          <input type="email" class="input" name="username" placeholder="Email">
        </div>
        <div class="form__group">
          <label for="" class="input__name">Пароль</label>
          <input type="password" class="input" name="password"
placeholder="пароль">
        </div>
      </div>
      <button class="form__button" type="submit">Реєстрація</button>
    </form>
  </div>
 </div>

 <%- include('partials/footer') %>

<%- include('partials/header') %>

 <div class="home">
  <div class="home__wrapper">
    <h1 class="home__header">Віддаленний моніторинг <br>систем крапельного
поливу</h1>
    <div class="home__links">
      <a href="/register" class="home__link">Реєстрація</a>
      <a href="/login" class="home__link">Вхід</a>
    </div>
  </div>
 </div>

<%- include('partials/footer') %>
```

```
const list = document.querySelectorAll(".nav-link");

const current = document.getElementById("current")
const field = document.getElementById("field")
const notification = document.getElementById("notification")
// const exsport = document.getElementById("exsport")

const currentDiv = document.querySelector(".current__wrapper");
const fieldDiv = document.querySelector(".sensors__date");
const notDiv = document.querySelector(".history__wrapper");
let canvas = window.document.querySelector('canvas');
let context = canvas.getContext('2d');
let chart = null;
let pauseMode = false;
const realTimeDemo = (xData, yData, data) => {
  let i = 5;
  let interval = setInterval(() => {
    if (i > data.length) clearInterval(interval);
    else if (!pauseMode) {
      chart.config.data.labels.push(xData[i]);
      chart.config.data.datasets[0].data.push(yData[i]);
      chart.config.options.scales.x.min++;
      chart.config.options.scales.x.max++;
      chart.update();
      i++;
    }
  }, 6000);
}
```

```
const createLineChart = (xData, yData) => {
  let gradient = context.createLinearGradient(0, 0, 0, window.screen.width / 2);
  gradient.addColorStop(0, 'rgba(74, 169, 230, 0.8)');
  gradient.addColorStop(1, 'rgba(74, 169, 230, 0.001)');
  let data = {
    labels: xData,
    datasets: [{
      label: 'Global Price of Aluminum',
      data: yData,
      pointStyle: false,
      fill: true,
      backgroundColor: gradient,
      borderWidth: 2,
      borderColor: 'rgba(74, 169, 230, 1)',
      tension: 0.2
    }]
  }
  let xScaleConfig = {
    min: 0,
    max: 5,
    ticks: {
      autoSkip: true,
      maxRotation: 0,
      // minRotation: 90,
      color: 'rgba(74, 169, 230, 0.9)'
    },
    border: {
      color: 'rgba(74, 169, 230, 1)'
    },
    grid: {
```

```
    color: 'rgba(74, 169, 230, 0.3)'
  }
}
let yScaleConfig = {
  min: 0,
  max: 100,
  ticks: {
    color: 'rgba(74, 169, 230, 0.9)'
  },
  border: {
    color: 'rgba(74, 169, 230, 1)'
  },
  grid: {
    color: 'rgba(74, 169, 230, 0.3)'
  }
}
let config = {
  type: 'line',
  data: data,
  options: {
    scales: {
      x: xScaleConfig,
      y: yScaleConfig
    },
    plugins: {
      legend: {
        display: false
      }
    },
    animation: {
```

```
        duration: 400,
        easing: 'linear'
      }
    }
  }
  chart = new Chart(context, config);
  chart.canvas.parentNode.style.height = '400px';
  chart.canvas.parentNode.style.width = '900px';
}
axios.get('data.json')
  .then((response) => {
    let data = response.data.data;
    let xData = [];
    let yData = [];
    for (let i = data.length - 1; i > 0; i--) {
      if (data[i].value !== '.') {
        xData.push(data[i].date);
        yData.push(data[i].value);
      }
    }
    let xStartData = [];
    let yStartData = [];
    let xParseData = [];
    let yParseData = [];
    for (let i = 0; i < data.length; i++) {
      if (i < 50) {
        xStartData.push(xData[i]);
        yStartData.push(yData[i]);
      } else {
        xParseData.push(xData[i]);
```

```
        yParseData.push(yData[i]);
      }
    }
    createLineChart(xStartData, yStartData);
    realTimeDemo(xParseData, yParseData, data);
  });

window.addEventListener('click', () => pauseMode = !pauseMode);

list.forEach((listItem) => {
  listItem.addEventListener("click", (event) => {
    list.forEach(list1 => {
      list1.classList.remove("active")
    })
    console.log(event.target)
    event.target.classList.add("active")
  })
})

current.addEventListener("click", () => {
  currentDiv.style.display = "block";
  fieldDiv.style.display = "none";
  notDiv.style.display = "none"
});

field.addEventListener("click", () => {
  currentDiv.style.display = "none";
  fieldDiv.style.display = "flex";
  notDiv.style.display = "none"
});
```

```
notification.addEventListener("click", () => {
  currentDiv.style.display = "none";
  fieldDiv.style.display = "none";
  notDiv.style.display = "block"
})
<%- include('partials/header') %>

  <div class="wrapper__main">
    <nav>
      <div class="sidebar">
        <div class="logo">
          <span class="logo-name">Drip</span>
        </div>
        <div class="sidebar-content">
          <ul class="lists">
            <li class="list">
              <a href="#" id="current" class="nav-link active">
                <i class="bx bx-home-alt icon"></i>
                <span class="link">Загальне</span>
              </a>
            </li>
            <li class="list">
              <a href="#" id="field" class="nav-link">
                <i class="bx bx-bar-chart-alt-2 icon"></i>

                <span class="link">Поле</span>
              </a>
            </li>
            <li class="list">
```

```
            <a href="#" id="notification" class="nav-link">
              <i class="bx bx-bell icon"></i>
              <span class="link">Свовіщення та історія</span>
            </a>
          </li>
        </ul>
        <div class="bottom-cotent">
          <li class="list">
            <a href="#" class="nav-link">
              <i class="bx bx-log-out icon"></i>
              <span class="link">Вихід</span>
            </a>
          </li>
        </div>
      </div>
    </div>
  </nav>
 </div>
 <div class="main__wrapper">
  <div class="current__wrapper">
    <div class="info">
      <div class="pumb__wrapper">
        <label class="power">
          <input type="checkbox">
          <div>
            <svg viewBox="0 0 44 44">
              <path d="M22,6 C31,6 38,13 38,22 C38,31 31,38 22,38 C13,38 6,31 6,22
C6,13 13,6 22,6 L22,28" id="path">
              </path>
            </svg>
```

```
        </div>
      </label>

      <!-- dribbble -->
      <a class="dribbble" href="https://dribbble.com/shots/5255048-On-Off-Toggle"
target="_blank"><img
          src="https://cdn.dribbble.com/assets/dribbble-ball-1col-dnld-
e29e0436f93d2f9c430fde5f3da66f94493fdca66351408ab0f96e2ec518ab17.png"
          alt=""></a>
      <h2 class="pump__header">Зупинити автоматичний полив </h2>
    </div>
    <div class="info__date">
      <div class="soil__moisture wrap">
        <i class="fas fa-tint fa-temperature-half ico"></i>
        <h2 class="soil__header">Вологість ґрунту</h2>
        <p class="soil__value">70%</p>
      </div>
      <div class="air__temperature wrap">
        <i class="fa-solid fa-temperature-half ico"></i>
        <h2 class="soil__header">Температура повітря</h2>
        <p class="soil__value">
          <%= data %>&#8451;
        </p>
      </div>
      <div class="air__humidity wrap">
        <i class="fas fa-tint fa-temperature-half ico"></i>
        <h2 class="soil__header">Вологість повітря</h2>
        <p class="soil__value">
          <%= data1 %>%
        </p>
```

```
      </div>
    </div>
  </div>
  <div class="canvas__wrapper">
    <h4 class="name__chart">Зміна вологості ґрунту в
      залежності від часу</h4>
    <canvas></canvas>
  </div>
</div>

<div class="sensors__date">

  <div class="one__area">
    <p class="sensor__date">
      <%=sensors[0]%>%
    </p>
    <p class="sensor__date pora1">
      <%=sensors[1]%>%
    </p>
    <p class="sensor__date pora1">
      <%=sensors[2]%>%
    </p>
  </div>
  <div class="two__area">
    <p class="sensor__date ">
      <%=sensors[3]%>%
    </p>
    <p class="sensor__date pora">
      <%=sensors[4]%>%
    </p>
```

```
        <p class="sensor__date">
          <%=sensors[5]%>%
        </p>
      </div>
      <div class="three__area">
        <p class="sensor__date pora1">
          <%=sensors[6]%>%
        </p>
        <p class="sensor__date">
          <%=sensors[7]%>%
        </p>
        <p class="sensor__date pora1">
          <%=sensors[8]%>%
        </p>
      </div>
    </div>
    <div class="history__wrapper">
      <div class="history__container">
        <h4 class="history__title">Полив <span
style="color:red">вимкнено</span></h4>
        <p class="histry__date">Вівторок Травень 16 2023 17:02</p>
      </div>
      <div class="history__container">
        <h4 class="history__title">Полив <span
style="color:green">ввімкнено</span></h4>
        <p class="histry__date">Вівторок Травень 16 2023 15:30</p>
      </div>
      <div class="history__container">
        <h4 class="history__title">Полив <span
style="color:red">вимкнено</span></h4>
```

```
      <p class="histry__date">Вівторок Травень 16 2023 14:50</p>
     </div>
     <div class="history__container">
      <h4 class="history__title">Полив <span
style="color:green">ввімкнено</span></h4>
      <p class="histry__date">Вівторок Травень 16 2023 14:02</p>
     </div>
     <div class="history__container">
      <h4 class="history__title">Полив <span
style="color:red">вимкнено</span></h4>
      <p class="histry__date">Вівторок Травень 16 2023 12:59</p>
     </div>
     <div class="history__container">
      <h4 class="history__title">Полив <span
style="color:green">ввімкнено</span></h4>
      <p class="histry__date">Вівторок Травень 16 2023 11:46</p>
     </div>
     <div class="history__container">
      <h4 class="history__title">Полив <span
style="color:red">вимкнено</span></h4>
      <p class="histry__date">Вівторок Травень 16 2023 10:02</p>
     </div>
     <div class="history__container">
      <h4 class="history__title">Полив <span
style="color:green">ввімкнено</span></h4>
      <p class="histry__date">Вівторок Травень 16 2023 9:11</p>
     </div>

   </div>
```

```
  </div>

  <%- include('partials/footer') %>

<script src ="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script src ="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/chartjs-plugin-zoom/dist/chartjs-plugin-zoom.min.js"></script>
<script src="script.js" type="module"></script>
</body>
</html>
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Drip</title>
  <link rel="stylesheet" href="css/styles.css">
  <link href="https://unpkg.com/boxicons@2.1.2/css/boxicons.min.css" rel="stylesheet" />
  <!-- Fonts -->
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css">
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link href="https://fonts.googleapis.com/css2?family=Josefin+Sans:wght@300;700&display=swap" rel="stylesheet">
  <script src="https://kit.fontawesome.com/245434d094.js"
```

```
crossorigin="anonymous"></script>
</head>

<body>
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Пояснювальні документи	
Диплом_Галета.doc	Пояснювальна записка Документ Word.
Диплом_Галета.pdf	Пояснювальна записка форматі PDF
Програма	
ProgramHaleta.rar	Архів. Містить коди откомпільовану програму
Презентація	
Презентація_Галета.ppt	Презентація проекту