Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)
Програмна система для автоматизації процесів догляду за промисловим садом
(тема)

Виконав:
здобувач 4 року навчання
групи ПЗПІ-21-8
Герман АРТУШЕВСЬКИЙ
(Власне ім'я, ПРІЗВИЩЕ)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. кафедри ПІ Ірина ЛЕЩИНСЬКА
(посада, Власне ім'я, ПРІЗВИЩЕ)

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

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Артушевському Герману Володимировичу
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система для автоматизації процесів догляду за промисловим садом

Затверджена наказом по університету від 19.05.2025 № 397 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 20.06.2025

3. Вихідні дані до роботи Розробити програмний застосунок, котрий буде містити в собі серверну частину, спосіб комунікації із IoT девайсами та додаток для Windows.

4. Перелік питань, що потрібно опрацювати в роботі: Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень,тестування розробленого програмного забезпечення, висновки, додатки

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	09.04.2025	*виконано*
2	Створення специфікації ПЗ	11.04.2025	*виконано*
3	Проектування ПЗ	14.04.2025	*виконано*
4	Розробка ПЗ	27.04.2025	*виконано*
5	Тестування ПЗ	03.05.2025	*виконано*
6	Оформлення пояснювальної записки	11.05.2025	*виконано*
7	Підготовка презентації та доповіді	15.05.2025	*виконано*
8	Попередній захист	02.06.2025	*виконано*
9	Нормоконтроль, рецензування	05.06.2025	*виконано*
10	Здача роботи у електронний архів	06.06.2025	*виконано*
11	Допуск до захисту у зав. кафедри	14.06.2025	*виконано*

Дата видачі завдання «8» «квітня» 2025р.

Здобувач _____
(підпис)

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

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра, 71 стор., 33 рис., 1 табл., 14 джерел.

АВТОМАТИЗАЦІЯ ПРОЦЕСІВ, ПРОГРАМНА СИСТЕМА, ПРОМИСЛОВИЙ САД, C#, JETBRAINS RIDER, .NET, UNITY3D

Об’єкт розробки – програмна система для автоматизації робочих процесів.

Мета розробки – створення програмної системи, яка зможе автоматизувати процеси догляду за промисловим садом.

Метод рішення - Середовище розробки .Net Core та JetBrains Rider, мова програмування C#.

У результаті розробки було створено програмну систему для догляду за промисловим садом із можливостю налаштування автоматичних процесів та отримання звіту щодо стану саду.

ABSTRACT

AUTOMATION OF PROCESSES, SOFTWARE SYSTEM, INDUSTRIAL ORCHARD, C#, JETBRAINS RIDER, .NET, UNITY3D

The object of development – a software system for automating work processes.

The purpose of development – to create a software system capable of automating the maintenance processes of an industrial orchard.

Method of implementation – development environment: .NET Core and JetBrains Rider; programming language: C#.

As a result of the development, a software system was created for maintaining an industrial orchard, featuring the ability to configure automated processes and receive reports on the orchard’s condition.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

JSON - JavaScript Object Notation

HTTP – Hypertext Transfer Protocol

MVP – Model-View-Presenter

IOT – Internet of Things

MQTT - Message Queuing Telemetry Transport

ВСТУП

Сучасне сільське господарство активно впроваджує використання цифрових технологій задля покращення показників ефективності виробництва. Сучасні технології допомагають фермера зменшити вартість виробництва, знизити ризики та покращити якість продукції. Особливо це стосується садівництва, в якому догляд за насадженнями вимагає значних ресурсів, постійного моніторингу стану дерев та рутинного виконання агротехнічних заходів. Перераховані причини надають актуальність створенню програмної системи для автоматизації процесів догляду за промисловим садом.

Програмна система має на меті зменшити людський фактор та підвищити точність прийняття рішень щодо піклування за садом. Передбачається можливість отримати звіт щодо стану промислового саду, налаштувати автоматичні процеси, які забезпечать безперебійний догляд за станом рослин.

Метою роботи є створення програмної системи для автоматизації процесів догляду за промисловим садом, яка дозволяє здійснювати облік розумних пристроїв, отримувати звіт щодо стану виробництва та налаштовувати виконання автоматичних аграрних заходів. Основними задачами дослідження є аналіз вимог до системи, розробка її архітектури, реалізація ключових функцій та проведення повного тестування функціоналу системи.

Програмна система буде виконана із використанням таких технологій, як C#, .Net Core, JetBrains Rider, Unity, MQTT, SQL. Інноваційність рішення полягає у розробці системи, яка надає можливість легкого впровадження взаємодії із будь-яким розумними пристроям, що надає змогу користувачам налаштовувати різноманітні процеси в залежності від постійно змінних потреб.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Сучасний світ зробив розробку програмного забезпечення для аграрного сектор окремим напрямом у сфері інформаційних технологій. Дана галузь активно розвивається, пропонуючи рішення, що охоплюють доволі широкий спектр можливих завдань: від моніторингу стану ґрунтів до навігації техніки та планування сільськогосподарських робіт. Використання IoT пристроїв стає популярнішим із кожним днем, про що свідчить постійне збільшення частки ринку, яку займають компанії, які займаються розробкою розумних девайсів. Наразі найпопулярнішими сферами використання розумних технологій є системи розумного дому, системи розумного міста та медичний сектор, що означає, що є дуже великий простір для впровадження нових аграрних технологій [1].

Одним із найперспективніших напрямів цієї галузі є автоматизація догляду за багаторічними насадженнями, зокрема промисловими садами. Цей сегмент потребує спеціалізованих програмних систем, які здатні вести облік великої кількості пристроїв, враховувати стан виробництва, метеорологічні умови, а також надавати зручний інтерфейс керування.

З розвитком технологій IoT з’являється все більше пристроїв, які можна інтегрувати в систему автоматизації процесів догляду за промисловим садом. Це можуть бути сенсори вологості ґрунту, повітря, сенсори температури повітря, рівня освітлення, а також системи крапельного поливу. Усі ці пристрої можуть бути налаштовані задля автоматичного виконання різних задач, що суттєво зменшить витрати води, світла, підвищить врожайність та забезпечить сталість аграрних процесів.

На ринку вже існує певна кількість компаній, які частково вирішують перелічені проблеми. Цим займаються такі компанії, як Gardena, ClickAndGrow, ThingsBoard та інші. Однак більшість наявних рішень є комерційними, складними у впровадженні та надто дорогими для малих і середніх господарств. Також вони не завжди враховують специфічні локальні кліматичні умови або специфіку

вирощування конкретних культур. Таким чином, на ринку зберігається потреба в доступних, адаптованих до місцевих умов програмних системах, які дозволяють ефективно автоматизувати процеси догляду за промисловим садом. Це відкриває можливість створення нових, інноваційних рішень, які орієнтовані на простоту використання, легку масштабованість та інтеграцію із сучасними технологіями.

1.2 Виявлення та вирішення проблем

У процесі аналізу предметної галузі було виявлено низку проблем, які заважають ефективній роботі та ускладнюють догляд за промисловим садом. Основні з них пов’язані із відсутністю можливості простого додавання нових пристроїв до існуючої структури виробництва та складністю впровадження. Програмна система для автоматизації процесів догляду за промисловим садом створена для вирішення наступних проблем:

- надто висока вартість комплектуючих для малих та середніх підприємств;
- складність додавання нових пристроїв до системи;
- відсутність простого та одночасно зручного інтерфейсу для керування автоматизацією процесів;
- великий вплив людського фактору на ефективність виробництва;

Метою розробки програмної системи для автоматизації процесів догляду за промисловим садом є створення сервісу, який допоможе користувачам отримати доступ до недорогого продукту із такими функціями, як: адміністрування промислового саду, отримання звітів від розумних пристроїв, ведення обліку розумних пристроїв, налаштування автоматичних процесі.

Адміністрування промислового саду надає користувачу можливість надати доступ до управління програмної системи іншим користувачам, що є обов’язковою потребою для промислового виробництва. Отримання звітів від розумних пристроїв надає змогу моніторингу стану виробництва. Облік розумних пристроїв допомагає користувачу управляти девайсами, які мають доступ до відправки звітів. Впровадження простого інтерфейсу керування автоматизацією прибирає потребу у довгому навчанні задля вирішення нескладних проблем. Також, налаштування

автоматизації процесів прибирає людський фактор, який впливає на ефективність роботи промислового саду.

1.3 Визначення аналогів

Одним із найбільших аналогів у сфері розробки розумних пристроїв для автоматизації рутинних процесів догляду за садом є сервіс ClickAndGrow [2]. Головними перевагами сервісу є дуже легке впровадження та широкий спектр послуг від можливості придбати самі розумні пристрої до можливості придбати насадження, які найкраще підійдуть саме для них. Веб застосунок надає можливість передивитись список усіх доступних товарів, замовити їх та отримати допомогу у разі будь-яких питань (див. рис. 1.1).

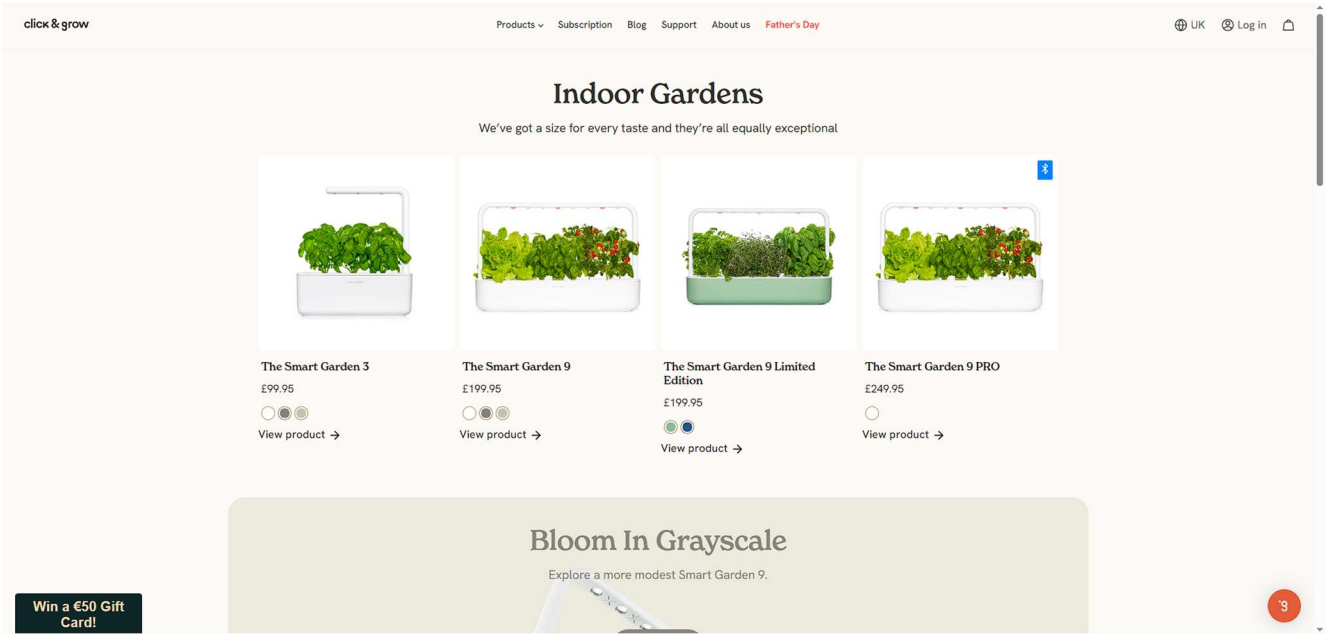

Рисунок 1.1 – Головна сторінка сайту ClickAndGrow (за даними [2])

Мінусами цього сервісу є дуже велика вартість продуктів та відсутність рішень для великих площ насаджень, що ускладнює його використання малими та середніми виробництвами. Найдешевший пристрій для вирощення рослин коштує від 100 євро та надає змогу вирощувати лише 3 рослини одночасно, що сильно збільшує вартість саду. Найкраще ClickAndGrow підходить для невеликих

закритих садів, які мають на меті вирощувати невеликі рослини, сервіс надає змогу дуже точно та надійно відслідковувати стан насаджень та автоматизувати такі рутинні задачі, як полив та слідкування за освітленням.

Іншим великим конкурентом є сервіс Gardena [3]. Він надає широкий спектр продуктів для вирощення та підтримки промислового саду. Одним із основних його продуктів є системи для автоматичного поливу та догляду за ґрунтом. Також сервіс надає користувачу доступ до обширної бібліотеки знань, які допомагають впровадити їх рішення. Веб застосунок дозволяє переглядати список доступних товарів, отримати допомогу у разі виникнення питань та отримати доступ до інструкцій щодо використання сервісу (див. рис. 1.2).

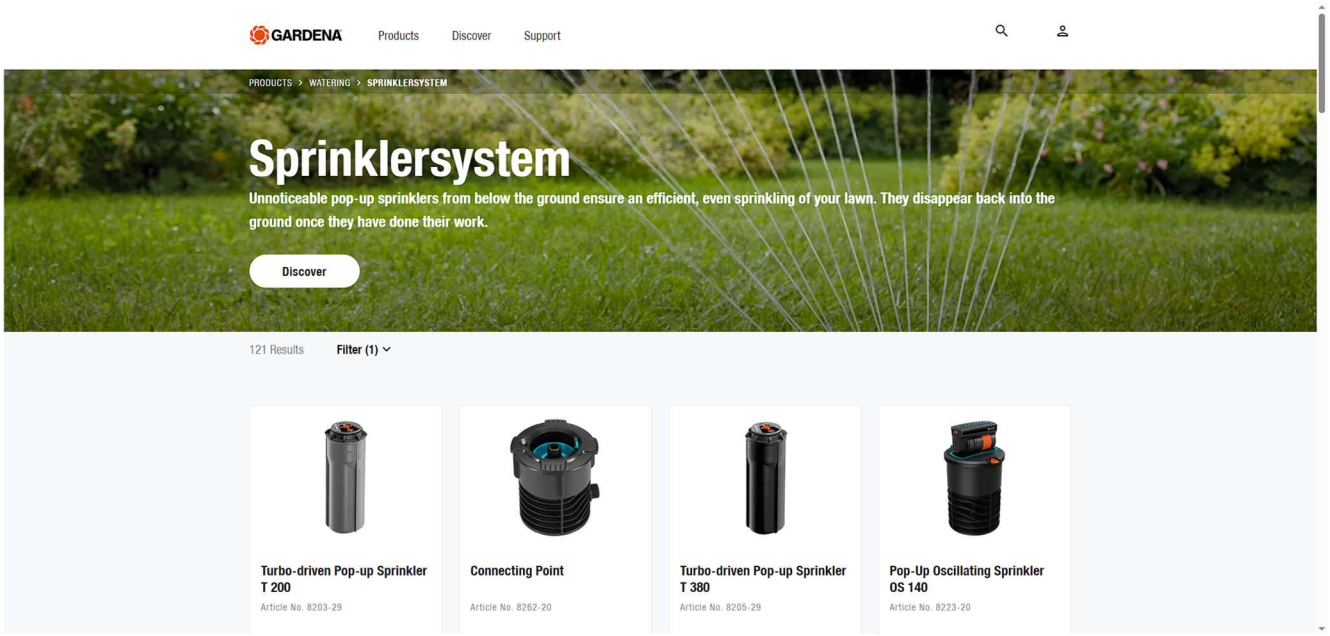

Рисунок 1.2 – Головна сторінка сервісу Gardena (за даними [3])

Іншим великим аналогом, який створює пристрої, що допомагають автоматизувати процеси догляду за садом є ThingsBoard [4]. Це великий сервіс, який пропонує розумні пристрої для обширної кількості сфер виробництва. Однією із цих сфер є вирощення агрокультур, для якої в них є широкий спектр товарів, від невеликих систем поливу до аналізу погодних умов за допомогою штучного інтелекту. Також, ThingsBoard надає користувачам доступ до великої бази знань

задля швидкого впровадження їх сервісу. Веб застосунок надає доступ до їх продуктів, підтримки користувачів, бази знань та прикладів щодо використання їх систем (див. рис. 1.3).

Рисунок 1.3 – Головна сторінка ThingsBoard (за даними [4])

Усі названі сервіси мають свої переваги та недоліки, які впливають на можливість їх використання для малих та середніх промислових садів. Усі вони назначені для різних вимог щодо виробництва та краще підходять для тих, чи інших промислових садів.

Під час аналізу аналогів було визначено переваги, які найбільше приваблюють користувачів:

- простота впровадження;
- доступ до бази знань;
- широкий спектр IoT пристроїв;
- зручний інтерфейс для налаштування пристроїв;
- можливість придбати готові насадження для пристроїв;
- рішення для аналітики погодних умов.

Окрім переваг сервіси-аналоги також мають певні недоліки, які ускладнюють використання їх продуктів для малих та середніх промислових садів.

У таблиці 1.1 наведено детальний аналіз переваг та недоліків сервісів-аналогів.

Таблиця 1.1 – Аналіз переваг та недоліків аналогів (виконано самостійно)

Назва аналогу	Опис	Переваги	Недоліки
ClickAndGrow	Сервіс надає доступ для розумних пристроїв, що допомагають вирощувати рослини у закритих садах	- просте впровадження; - підходить для закритих садів; - можливість управління кожною окремою рослиною	- велика вартість пристроїв; - відсутність рішень для великих садів; - можлива автоматизації лише поливу та освітлення
Gardena	Сервіс надає доступ до пристроїв автоматичного поливу та уходу за ґрунтом	- якість пристроїв; - широк спектр пристроїв для різних потреб; - можливість автоматичного догляду за ґрунтом	- складність впровадження; - недостатня кількість документації; - відсутність змоги моніторингу стану саду.
ThingsBoard	Сервіс, що допомагає впровадити розумні пристрої для промислових садів	- аналіз погодних умов; - доступ до open-source версії; - обширна база знань; - високий рівень інтеграції	- відсутність готових пристроїв; - впровадження потребує глибоких знань;

Дані, продемонстровані в таблиці, показують, що немає одного сервісу, який задовільнив би усі потреби малих та середніх промислових садів. Деякі сервіси мають надто високу вартість впровадження в той час, як інші потребують глибоких знань задля їх впровадження. Ринок має потребу у новій програмній системі, що створить можливість дешево та легко впровадити інфраструктуру для автоматизації процесів догляду за промисловим садом.

1.4 Постановка задачі

Аналіз сфери розумних пристроїв для промислових садів надає змогу зрозуміти, що зрозуміти потреби користувачів щодо ефективного сервісу. Головна мета цього проекту, це розробка програмної системи, яка зможе задовільнити потреби користувачів щодо автоматизації рутинних задач догляду за промисловими садами малого та середнього розміру. Задля забезпечення потреб користувачів програмна система має враховувати такі можливості:

- надавання доступу до налаштування автоматизованих задач робітникам;
- видалення доступу до налаштування автоматизованих задач робітникам;
- перегляд доступних IoT пристроїв для саду, які можуть відправляти дані щодо промислового саду;
- додавання доступних пристроїв до саду;
- видалення пристроїв із переліку;
- отримання звітів щодо температури саду;
- отримання звітів щодо відносного рівня вологи в саду;
- отримання звітів щодо рівня освітлення саду;
- налаштування автоматичного управління освітленням;
- налаштування автоматичного поливу;
- створення власного процесу користувача;
- редагування процесів автоматизації;
- видалення процесів автоматизації;
- зменшення людського фактору щодо задач догляду за садом;
- тестування та вдосконалення системи;

Таким чином, програмна система для автоматизації процесі догляду за промисловим садом буде мати можливості задоволення потреб різних користувачів. Оскільки, сервісом можуть користуватись різні люди із різними цілями, виникає потреба в аналізі цільової аудиторії. Цільова аудиторія розроблюваної системи враховує різні групи користувачів, із різними потребами і цілями. Це такі групи, як:

- власники невеликих закритих садів, яким потрібна можливість недорого та легко інтегрувати інфраструктуру для автоматизації задач;
- власники невеликих полів, яким потрібна можливість автоматизувати задачі щодо поливу на певній площі;
- власники середніх промислових садів, які потребують можливості відслідковування змін у стані саду та налаштування великої кількості різних задач;
- люди, яким потрібно налаштувати рутинні задачі догляду за їх придомовою ділянкою;
- люди, які хочуть вирощувати невеликі рослини у рамках квартири;
- люди, яким потрібна інфраструктура, в яку можливе легке додавання нових пристроїв.

Цільова аудиторія програмної системи для автоматизації процесів догляду за промисловим садом складається із доволі великої кількості різних груп користувачів, що забезпечує популярність сервісу. Різноманітна цільова аудиторія потребує, щоб функціонал додатку задовольняв кожну із них, що створює потребу у детальному аналізі архітектури. Також, висока кількість користувачів створює більшу кількість навантаження на програмну систему, що, в свою чергу, викликає потребу у масштабуванні додатка та ефективній роботі із даними. Окрім цього, ефективна робота із даними може знизити вартість сервісу та збільшити ефективність роботи. Зменшення вартості сервісу надає змогу більшій кількості користувачів використовувати його. Також, методи збільшення ефективності роботи із даними повинні враховувати те, що кількість користувачів постійно

збільшується та є постійна вимога щодо масштабування програмної системи задля підтримання її роботи.

Здатність масштабуватись та ефективно оброблювати дані, це також одні із основних вимог щодо програмної системи, оскільки відсутність цих елементів унеможливлює надавання сервісу для широкої аудиторії та втрачає користувачів. Задля забезпечення гарного досвіду використання сервісом у користувачів із кожної групи цільової аудиторії повинні виконуватись усі вимоги щодо програмної системи, а також бути реалізовані усі функції.

Таким чином, було визначено функціональні та нефункціональні вимоги програмної системи, проаналізовано цільову аудиторію сервісу, потреби цієї аудиторії та основні проблеми, які можуть виникнути.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Аналіз вимог

На основі проведеного аналізу предметної області та аналізу сервісів-аналогів, а також із урахуванням потреб різних цільових груп програмної системи, було сформульовано основні вимоги до функціоналу, надійності, інтерфейсу та розширювання, що будуть входити до першої версії. Наявність реалізації цих вимог забезпечить можливість використання додатку обширною групою користувачів та збільшить актуальність програмної системи.

2.2 Функціональні вимоги

Функціональними вимогами визначається, що саме має робити програмна система. Вони формуються на основі аналізу практичної діяльності у сфері управління промисловими садами, а також із урахуванням потреб користувачів. Нижче надано перелік основних можливостей, які має реалізовувати додаток:

а) ведення обліку доступних промислових садів:
   1) додавання садів до обліку;
   2) редагування даних садів;
   3) видалення садів;

б) ведення обліку працівників саду:
   1) додавання працівників;
   2) видалення працівників;

в) ведення обліку IoT девайсів:
   1) додавання доступних девайсів;
   2) видалення доступних девайсів;

г) отримання звітів від розумних пристроїв:
   1) отримання звіту щодо температури повітря;
   2) отримання звіту щодо відносного рівня вологості;
   3) отримання звіту щодо рівня освітлення саду;

д) налаштування автоматичних процесів:

1) налаштування умов для відправлення даних через протокол MQTT.

2.3 Нефункціональні вимоги

Задля забезпечення усіх потреб користувачів програмна система для автоматизації процесів догляду за промисловим садом має виконувати певні нефункціональні вимоги. Під час аналізу було виявлені наступні нефункціональні вимоги:

- надання надійності виконання. Користувачі мають бути впевнені, що програма надає доступ до актуальних даних та безперебійно виконує роботу;
- безпека даних. Користувачі мать бути впевнені, що дані зберігаються в захищеному вигляді та не можуть бути передані 3-тій стороні;
- допомого користувача. Користувачі мають бути в змозі отримати допомогу при виникненні проблем, пов’язаних із програмною системою;
- програмна система має підтримувати масштабованість.

Надійність виконання означає, що програмна система має функціонувати 24/7 та, у разі будь-яких помилок, відновлювати свою роботу у найкоротші терміни. Безпека даних гарантує користувачам надійність збереження усіх конфіденційних даних та відсутність ризиків їх передачі 3-тим лицям. Усі паролі користувачів мають зберігатись у захешованому вигляді, без можливості їх відновити, не знаючи пароля. Також, користувачі мають бути в змозі видалити свої дані у разі потреби.

2.4 Апаратні вимоги

Як і будь-яка програмна система, програма для автоматизації процесів догляду за промисловим садом має свої потреби задля правильного функціонування. Задля користування додатком для Windows, у користувача має бути комп’ютер із, що найменше, 4 гігабайтами оперативної пам’яті, 200 мегабайтами вільного місця на диску та підключення до інтернету. Для гарантування безперебійної роботи усіх розумних пристроїв, в них має бути безперебійне підключення до мережі.

3 АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Програмна система для автоматизації процесів догляду за промисловим садом, це велика та комплексна система, до якої входять багато компонентів, що, в свою чергу, взаємодіють між собою. Нижче наведено Use Case діаграму додатку (див. рис. 3.1).

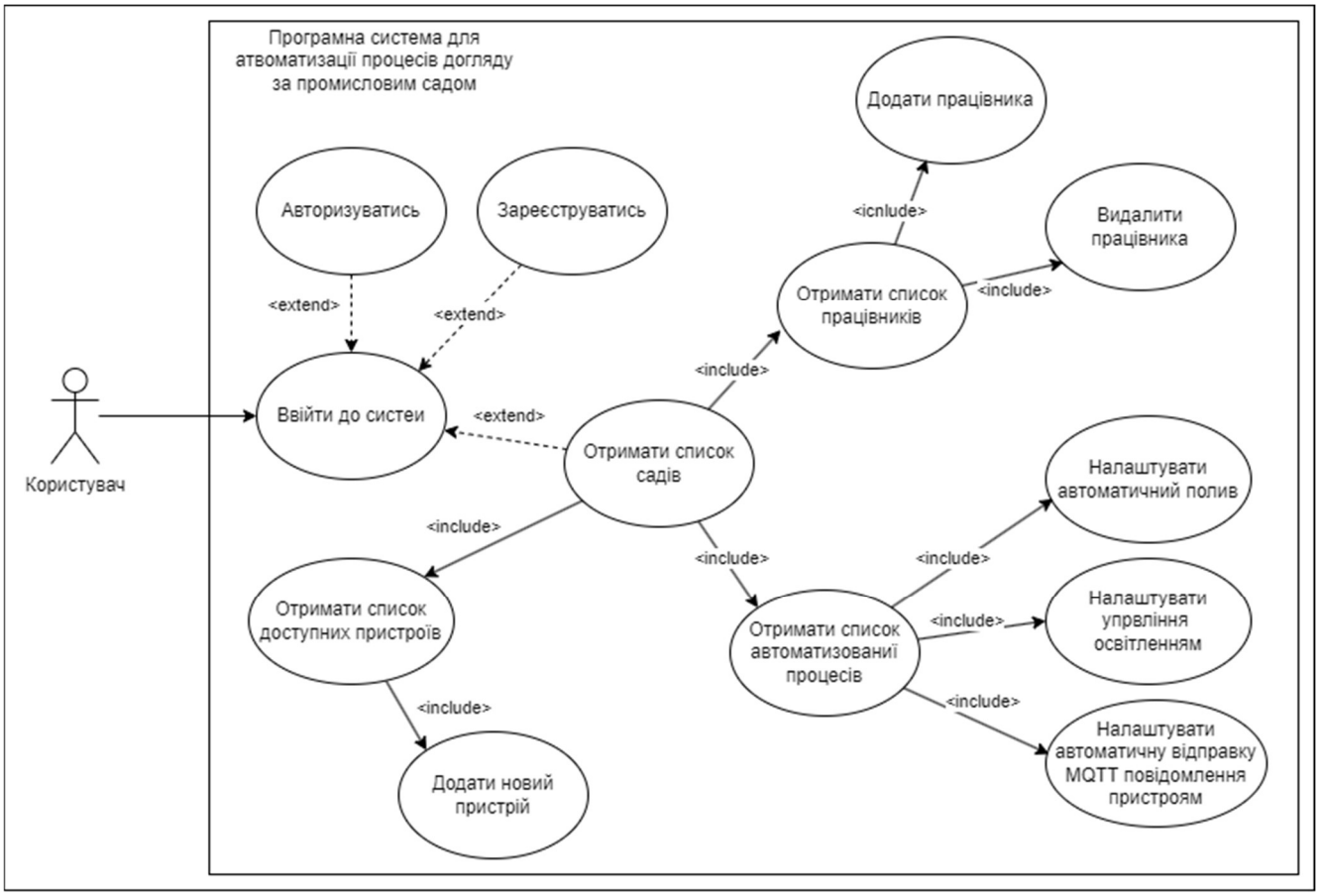

Рисунок 3.1 – Use Case діаграма (рисунок виконаний самостійно)

Продемонстрована Use Case діаграма показує усі можливості програми, які вона пропонує користувачу. Користувач в змозі зареєструватись, авторизуватись, переглядати та редагувати список доступних садів, переглядати та редагувати список працівників, переглядати та редагувати список доступних пристроїв, переглядати та редагувати список автоматизованих процесів.

Задля роботи програмної системи для автоматизації процесів догляду за промисловим садом, серверну частину додатку має бути розгорнуто на серверах, клієнт повинен мати додаток для Windows та брокер MQTT має працювати. Діаграму розгортання наведено нижче (див. рис. 3.2).

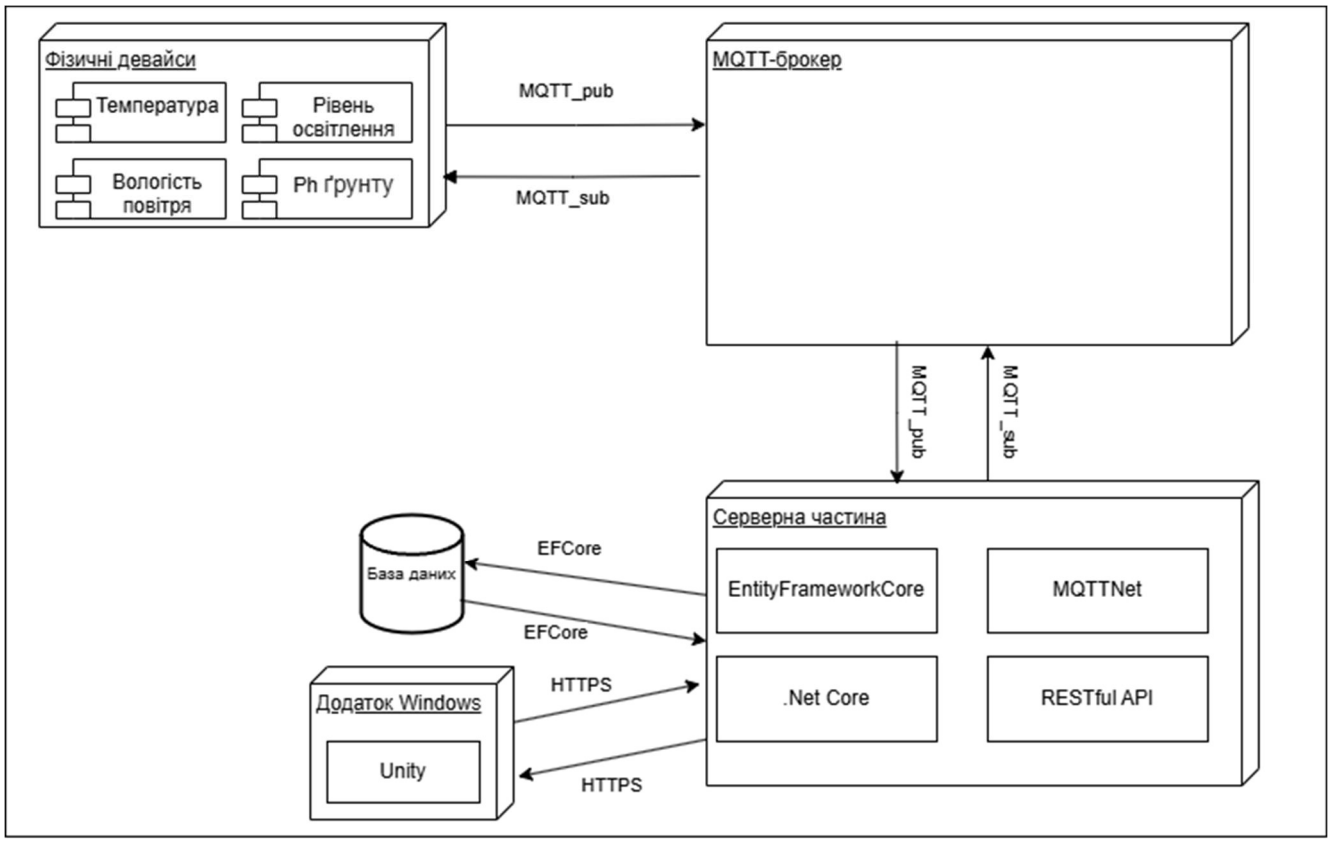

Рисунок 3.2 – Діаграма розгортання (рисунок виконаний самостійно)

Застосунок має такі компоненти, як: фізичні девайси, MQTT брокер, серверна частина, база даних та додаток Windows. Кожна з частин має працювати для того, щоб додаток був доступний користувачам.

Для взаємодій між клієнтом та сервером використовуються RESTful API та HTTPS [5]. Взаємодія між базою даних та серверною частиною гарантує EntityFramework Core, що надає змогу зручно використовувати різні бази даних та виконувати міграції даних, у разі потреби [6]. Задля комунікації між розумними пристроями та серверною частиною використовується протокол MQTT [7],

повідомлення надсилаються через сервіс-брокер, який забезпечує безперебійну доставку даних.

3.2 Проєктування структури зберігання даних

Задля зберігання даних використовується реляційна база даних MySql, запроси до якою роблять через програмне забезпечення EntityFramework Core, яке допомагає працювати із базами даних без залежності на специфічні формати запитів. Нижче приведено ER-діаграма бази даних (див. рис. 3.3)

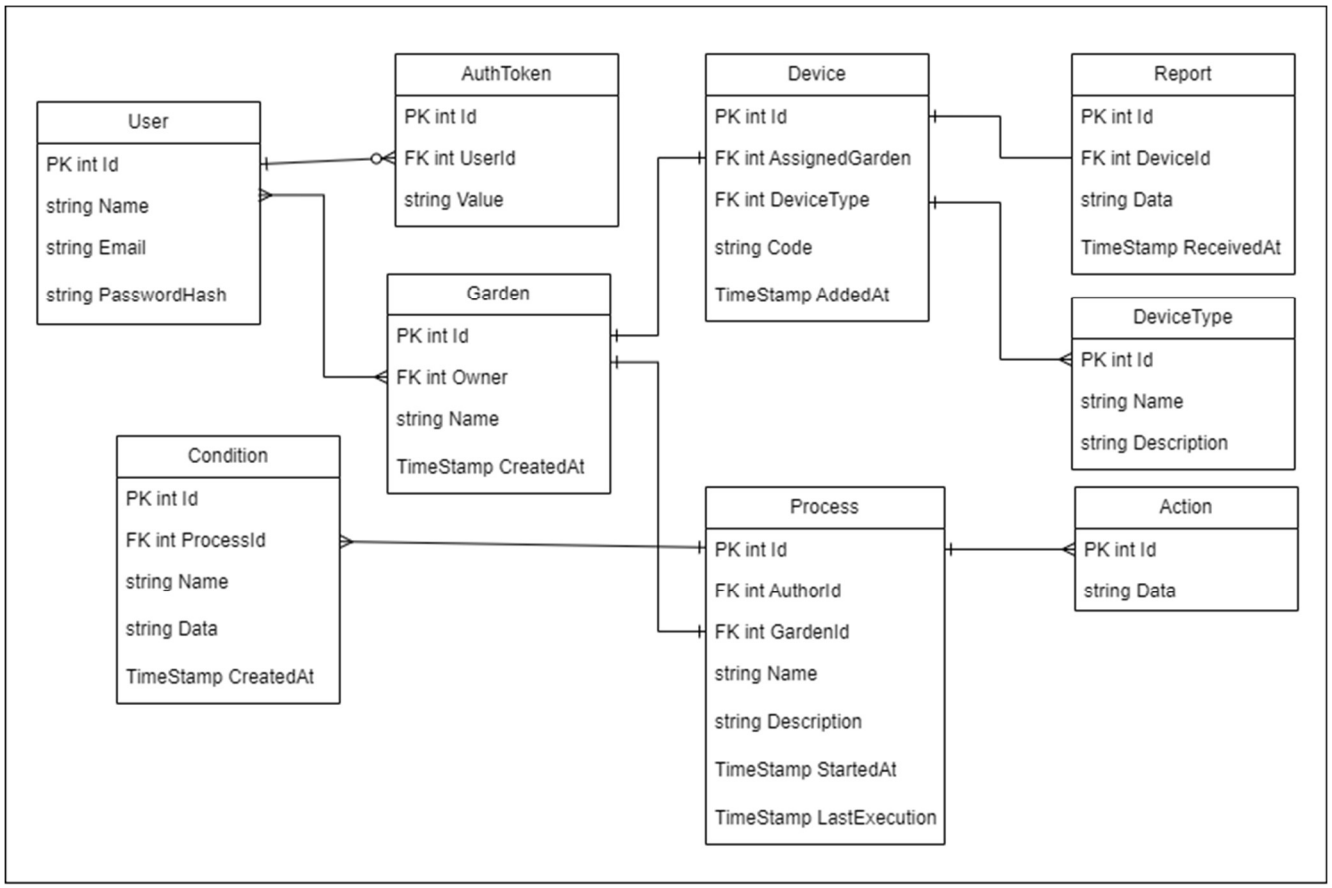

Рисунок 3.3 – ER-діаграма (виконаний самостійно)

База даних складається із продемонстрованих частин, які дозволяють користувачам створювати та адмініструвати облікові записи, записи щодо промислових садів, отримувати звіти від розумних пристроїв та адмініструвати автоматизовані процеси догляду за рослинами. Звіти від розумних пристроїв, умови виконання автоматичних процесів та дії, які вони мають виконати

зберігаються у форматі JSON, що надає змогу ефективно з ними працювати [8]. Задля роботи програмної системи, можливе використання будь-якої реляційної бази даних, що має можливості які задовольняють виникаючі у ході розробки потреби. Для першої версії було обрано базу даних MySQL, яка повністю відповідає потребам сервісу.

3.3 Проєктування архітектури ПЗ

Розробка програмної системи для автоматизації процесів догляду за промисловим садом побудована на архітектурі, яка передбачає наявність клієнта та сервера. Додаток для Windows виступає клієнтом в той час, як серверна частина відповідає за зберігання та обробку бізнес логіки, зберігання та управління даними та взаємодією із IoT частиною додатку. Саме такий підхід є найпоширенішим для веб розробки, яка дозволяє децентралізувати додаток та масштабувати його у разі потреби [9].

По-перше, основна із названих переваг, це відокремлення бізнес логіки від клієнтської частини, що надає змогу оновлювати її без втручання до клієнта та потреби постійного оновлення додатку для Windows. Також, відокремлення бізнес процесів сприяє забезпеченню їх безпечному зберігання та запобігає ситуаціям, коли треті лиця можуть отримати доступ до них.

Інша причина, чому варто використовувати саме цю архітектуру, це масштабованість. Під час постійної розробки програмно системи та збільшення кількості користувачів, виникає потреба у збільшенні серверів, які мають оброблювати запроси. У разі відсутності ефективного способу масштабування програмна система може не витримати навантаження, або нераціонально збільшувати витрати на підтримку, що, в свою чергу, зменшує її ефективність. Наразі, є велика кількість способів задля забезпечення масштабован
ості програмних систем, що полегшують їх розробку [10].

Наступна причина, чому варто обрати клієнт серверну архітектуру, це наявність IoT пристроїв та комунікації із ними. Усі дані та можливість підключитись до девайсів мають бути захищені, щоб не виникало можливості

використовувати їх людям, які не повинні мати доступ. Оскільки, усі дані передаються через окремий сервіс брокеру, можливості їх отримати без авторизації немає, що позбавляє загрози попадання даних до третіх лиць. Той факт, що усі дані, які відправляються до брокеру, відправляються лише сервером унеможливлює відправку неавторизованих повідомлень, що гарантує повну безпеку пристроїв.

Окрім переваг, клієнт-серверна архітектура має недоліки. Одним із найбільших недоліків такого способу створення програмних систем є потреба у постійній роботі сервера. Оскільки вся робота додатку залежить від роботи серверної частини, це збільшує ризики користувачів втратити доступ до сервісу.

Проаналізувавши усі переваги та недоліки клієнт-серверної архітектури програмних систем, можна сказати, що вона гарантує достатню ефективність розробки, надає можливість забезпечити безпечну роботу користувачів із сервісом та масштабувати додаток, у разі потреби.

Клієнтська частина побудована за допомогою програмної системи Unity. Такий вибір було зроблено через простоту розробки та оновлення додатка, легкість розробки інноваційного дизайну додатку та простий спосіб доставки додатка для Windows користувачам. Архітектура додатку побудована на MVP, це популярний шаблон проектування, який складається із моделі, відображення та доповідача [11]. Приклад роботи цього шаблону відображено на рисунку нижче (див. рис. 3.4)

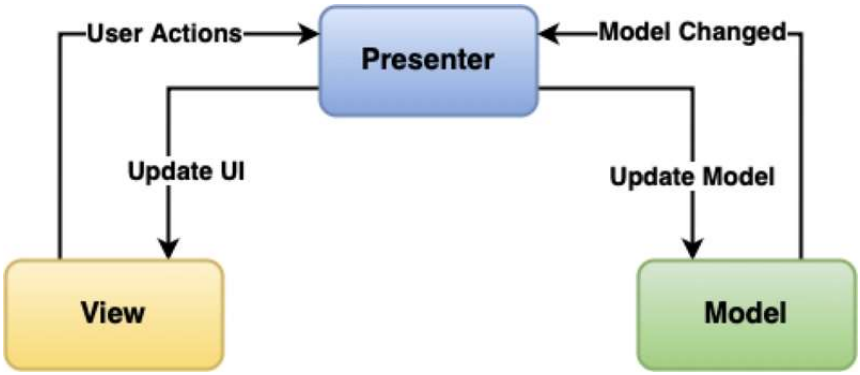

Рисунок 3.4 – Схема роботи шаблону MVP (за даними [11])

MVP, це похідний від MVC (Model-View-Controller) шаблон, що виник приблизно у 1990-х роках, який має на меті полегшити автоматичне тестування програмних систем та покращити відокремлення незалежних частин одна від одної.

Частина Model, це інтерфейс, який визначає дані для відображення у додатку. Вона може містити дані отримані із серверної частини у зручному форматі. Частина View відповідає за відображення, яким користується користувач. Вона відповідає за взаємодію користувача із програмною системою та правильним форматуванням даних. Частина Presenter являється посередником між іншими двома частинами та відповідає за правильну передачу даних між ними. Вона може реагувати на події у відображенні та змінювати дані моделей у разі потреби. Також, вона відповідає за доставку даних, якщо будь-яка частина моделі змінилась.

3.4 Приклади алгоритмів та методів

Бізнес-логіка програмної системи для автоматизації процесів догляду за садом має ефективно працювати із даними, які постійно надсилаються розумними пристроями та користувачами.

Оскільки, при успішній роботі сервісу, кількість користувачів постійно зростає, кількість даних, які зберігаються та обробляються на серверній частині, також постійно збільшується. Це призводить до виникнення потреби створення ефективних методів роботи із збереженням даних. Виникає потреба реалізації обробки даних на самих розумних пристроях, це допомагає зменшити кількість даних, які відправляються на сервера. Одним із способів зменшення кількості даних є реалізація функціоналу відправки повідомлень лише у разі зміни даних. У такому випадку IoT пристрої не посилають постійно дані щодо стану промислового саду, а лише повідомляють про зміни стану. Іншим прикладом обробки даних є конвертація даних із однієї обчислювальної системи до іншої, в залежності від налаштувань користувача. Одним із прикладів попередньої обробки даних на пристроях є зменшення шумів даних. Таким чином цей функціонал буде виконуватись на пристроях, а сервер буде лише зберігати ці дані.

Наприклад, для сенсорів температури, вологості повітря, рівня освітлення промислового саду може знадобитись прибирання короткочасних шумів та виділення тренду. Для вирішення цієї проблеми існує бага способів, одним із цих способів є використання функції SMA (Simple Moving Average), приклад якої наведено у вигляді формули 3.1.

$$SMA_t = \frac{1}{k} \sum_{i=t-k+1}^{t} x_i \, , t \geq k \tag{3.1}$$

де t – значення часу;

x – отримані дані з сенсору;

k – період ковзної середньої;

SMA_t – визначає функцію простої ковзної середньої;

$t \geq k$ – умова, що кількість значень має бути більша за період.

3.5 Створення UI/UX

Одним із найважливіших елементів програмної системи є інтерфейс, з яким взаємодіє користувач. Інтерфейс може приймати різну форму, від консольних команд, до програм із віртуальною реальністю. Більшість програмних систем використовують систему вікон, що передбачає зручний інтерфейс у вигляді блоків даних.

User Experience (UX), це одна із складових, що відіграє ключову роль у досвіді користувача при роботі із сервісом. UX охоплює душе широку область знань щодо Human Computer Interaction (HCI), та вивчає способи надання користувачам відповідний до їх потреб інтерфейс. Процес розробки включає в себе конкурентний аналіз, розробку шаблонів дизайну та вивчення функціонального сприйняття програмної системи. Це може включати в себе аналіз розташування елементів дизайну, впровадження інтерфейсу для людей із вадами здоров’я та інше. Існує чотири різні типи User Experience, що уявляють собою різні способи дослідження.

Перший тип, це дизайн взаємодії, він являє собою аналіз способів, якими користувач взаємодіє із програмною системою. Розуміння деталей, які потрібні користувачам, поведінки користувачів під час використання сервісу та мотивації користувачів щодо взаємодії із додатком допомагає розроблювати інтерфейси, з якими користувач хоче взаємодіяти. Наприклад, розроблюючи інтерфейс для взаємодії із звітами щодо промислового саду, потрібно розуміти, що хоче побачити користувач, яка інформація йому не потрібна та у якому вигляді він хоче її отримати.

Наступний тип, це візуальний дизайн, що являє собою вивчення способу відображення інтерфейсу для користувача. Це може включати в себе аналіз стилю додатку, аналіз вигляду програмних систем конкурентів та розробка інтерфейсу для людей із спеціальними потребами. Візуальний дизайн має на меті створення продукту, який приверне увагу користувача, посилить упізнавання бренду, буде викликати довіру у користувачів.

Інший тип, це інформаційна архітектура. Це включає в себе аналіз способів ефективного подання інформації користувачу. Гарна структура та організація даних допомагає більш ефективній взаємодії із користувачем.

Останній тип, це дослідження користувачів. Оскільки будь-яка програмна система розроблюється для користувачів, дуже важливо враховувати їх потреби. По-перше, треба визначити цільову аудиторію, яка буде користуватись сервісом. Дослідження дозволяє знайти нові способи взаємодії із користувачами, що, в свою чергу, допомагає покращити їх досвід [12].

User Interface (UI) дизайн визначає графічну структуру програмної системи. Це може включати в себе кнопки, тексти, зображення, відео та інші елементи. Також, дуже важливо визначити яким чином програмна система має реагувати на дії користувача. Гарний дизайн завжди відповідає актуальним даним, з якими працює користувач, та ніколи не створює відчуття нерозуміння. Користувацький інтерфейс має передавати цілісну естетику сервісу та виділятись посеред аналогів.

Проаналізувавши сервіси-аналоги можна зрозуміти, який функціонал потрібен користувачам, яким чином краще передавати інформацію та як користувачі взаємодіють із додатком.

Нижче приведено приклад візуального інтерфейсу додатку сервісу-аналогу ClickAndGrow (див. рис. 3.5).

Рисунок 3.5 – Інтерфейс налаштування процесів (рисунок зроблений самостійно)

Приведений вище рисунок надає змогу зрозуміти, який функціонал використовують користувачі сервісу-аналогу ClickAndGrow. Також, можна побачити в якому вигляді сервіс надає інформацію та способи управління даними.

Окрім цього, можна побачити кольорову схему, яка властива усім додаткам сервісу. Це є дуже важливим елементом, оскільки посилює розпізнання бренду.

Можна побачити, що важлива інформація виділена червоним кольором, що змушує користувача побачити її, це є дуже важливим елементом, оскільки покращує ефективність роботи користувача із додатком.

Також, в інтерфейсі присутні елементи, які надають спосіб адміністрування стану даних. Користувач може повернутись до стандартних налаштувань, відмінити зміни та зберегти їх. Це дуже важливо, оскільки зменшує ризик поганого досвіду користувача під час роботи.

Інший аналог, який варто досліджувати – це Things Board. Приклад інтерфейсу програмної системи аналога наведено нижче (див. рис. 3.6)

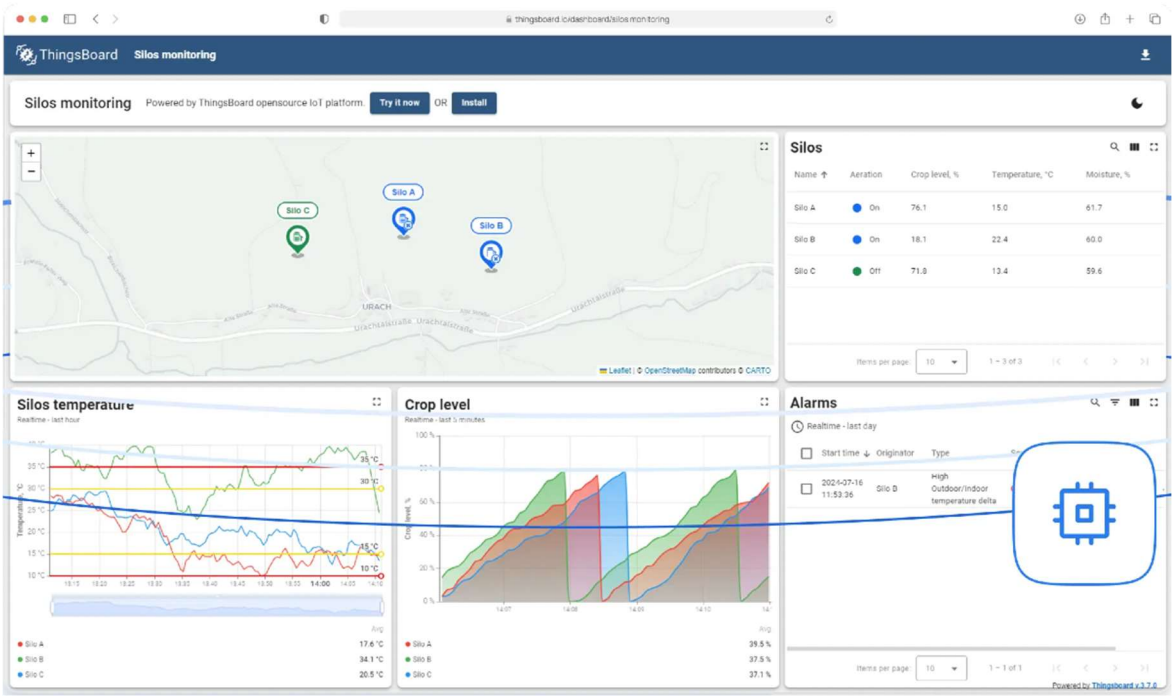

Рисунок 3.6 – Інтерфейс ThingsBoard (за даними [4])

На цьому зображенні можна побачити основні елементи, які відображаються для користувачів та формат в якому представлені дані. Також, можна побачити візуальний стиль додатку, який також посилює розпізнання бренду.

Сервіс має зручні відображення для звітів від розумних пристроїв, відображення стану пристроїв та невелике вікно налаштувань автоматичних процесів. Проаналізувавши графічні інтерфейси сервісів-аналогів можна побачити способи, якими інформація ефективно доставляється користувачу та способи, якими користувачі мають змогу налаштувати автоматичні процеси. Також, можна побачити, що усі звіти від пристроїв відображаються у вигляді графів.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Розробка відображення для клієнтської частини

Для розробки візуального інтерфейсу клієнтської частини було використано Unity. Цей вибір дозволяє легко створювати інноваційний дизайн для різних користувачів, які користуються сервісом. Для створення графіків та відображення використовувався фреймворк EzChart. За допомогою нього можливе створення графіків різних форматів. Нижче наведено приклад графіку (див. рис. 4.1).

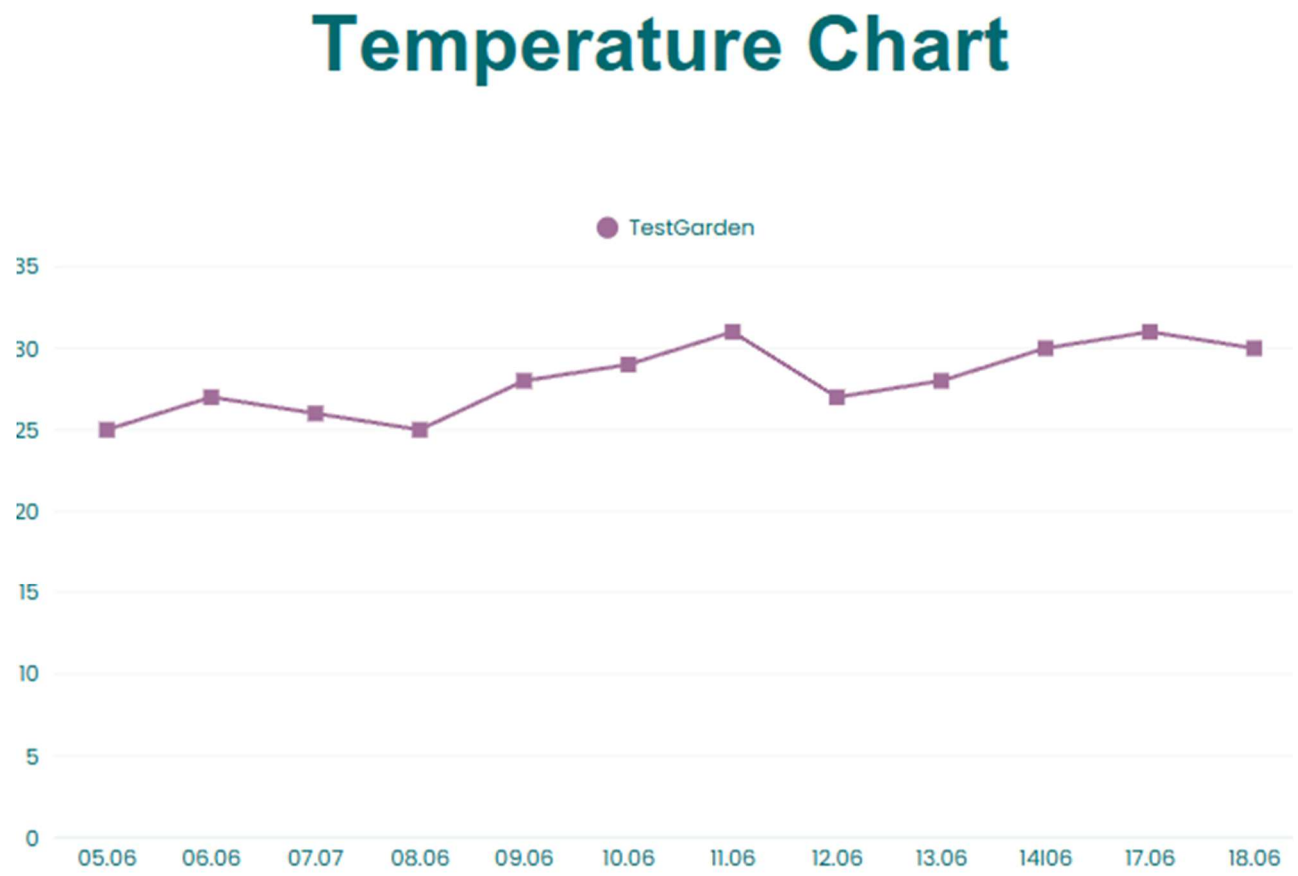

Рисунок 4.1 – Графік температури (рисунок виконано самостійно)

Цей графік відображає дані щодо температури у промисловому саду та надає змогу користувачам отримувати потрібні дані у зручному форматі. Також, підтримується взаємодія із графіком, користувач може детальніше переглядати будь-який із сегментів графіку та отримувати розгорнуту інформацію. Це надає

змогу розробити зручний інтерфейс та ефективніше працювати із даними на клієнтському додатку.

Відображення змінюється у реальному часі разом із отриманням даних від серверної частини. Можливе налаштування відображення та представлення графіку у різних форматах. Налаштування відображення графіку є дуже зручною функцією, що надає можливість користувачем налаштовувати роботу із даними так, як зручно їм. Це збільшує кількість користувачів, яким підходить додаток та надає змогу користувач із особливими потребами налаштувати додаток для зручної роботи із ним. Налаштування відображення графіку може враховувати в себе налаштування кольору графіку, налаштування формату відображення, налаштування формату даних, зо відображені на графіку, налаштовування інтерфейсу взаємодії із графіком, налаштування розміру шрифти тексту на графіку ті інше. Нижче наведено приклад взаємодії користувача із графіком (див. рис. 4.2).

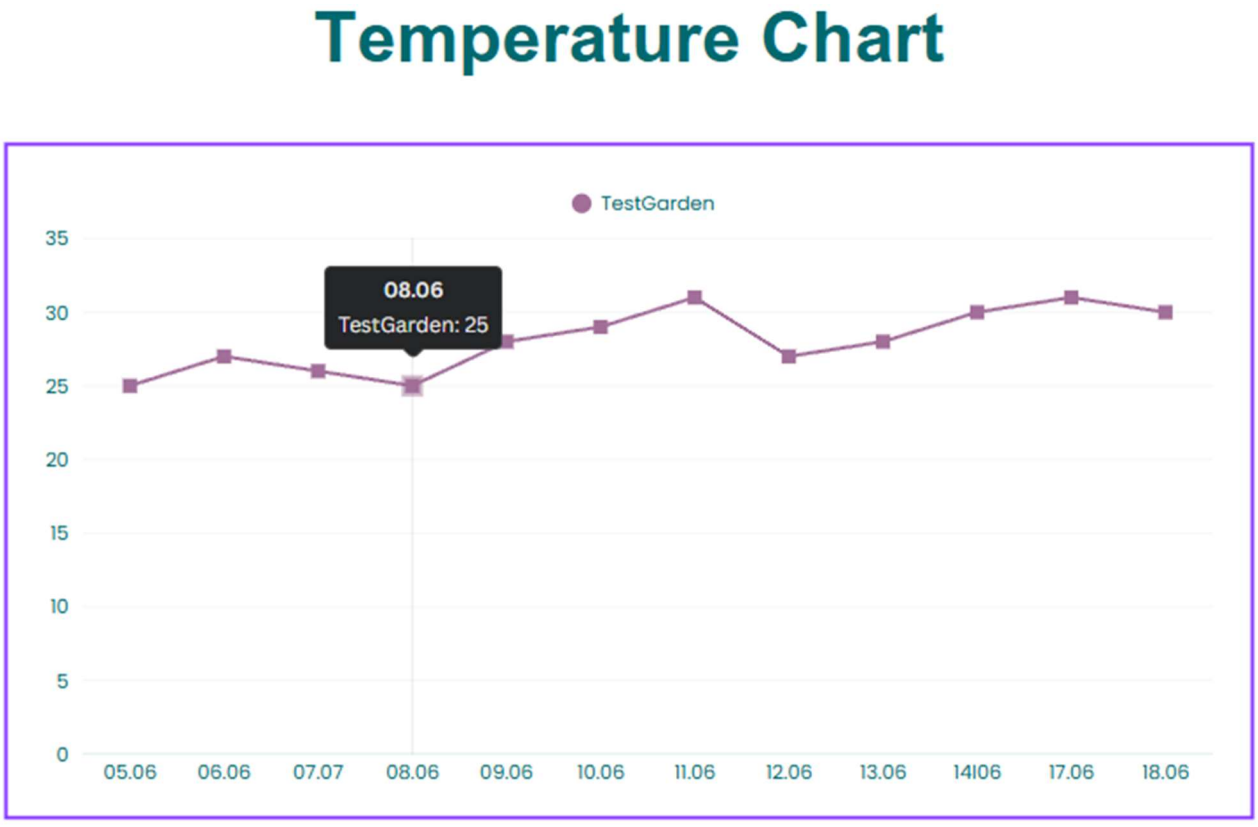

Рисунок 4.2 – Приклад взаємодії із графіком (рисунок виконано самостійно)

Також можна побачити приклад відображення графіку у іншому форматі (див. рис. 4.3).

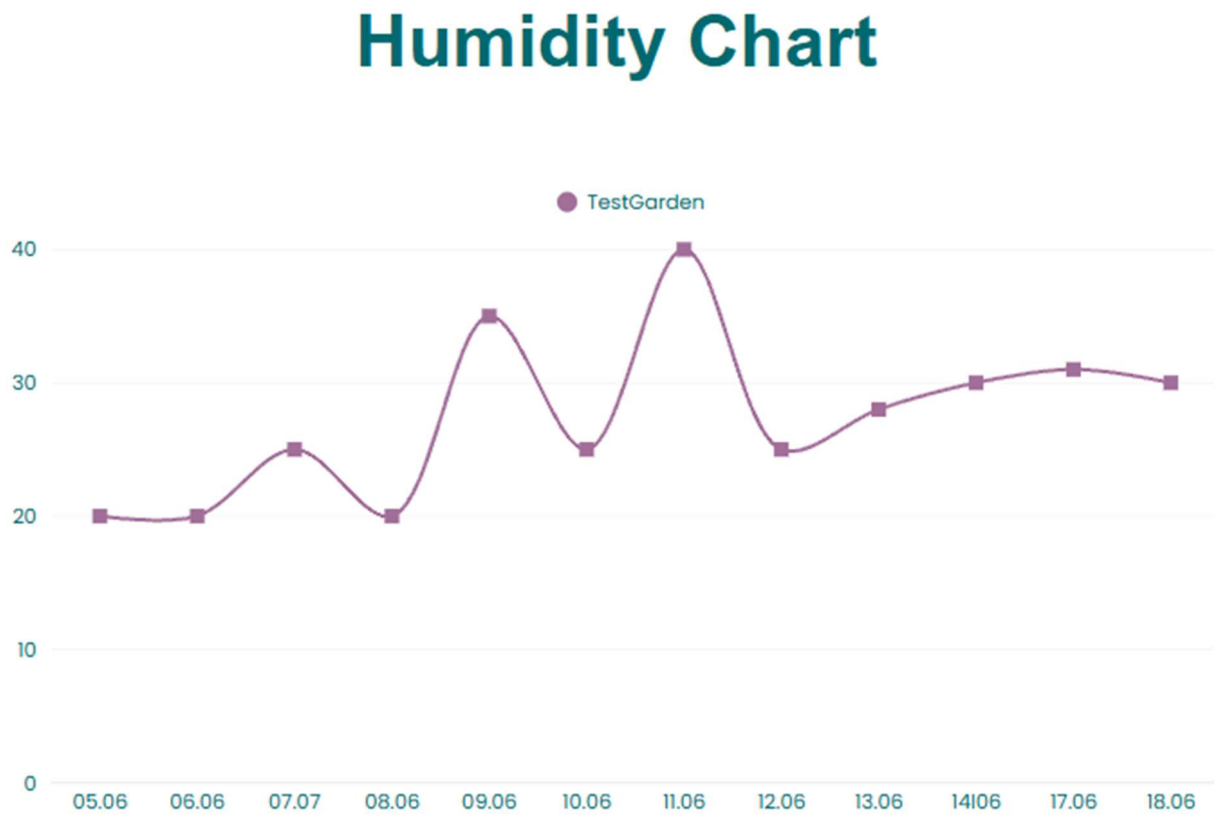

Рисунок 4.3 – Приклад іншого формату (рисунок виконано самостійно)

Задля ефективного відображення кругових діаграм, було створено код шейдеру для Unity. Він використовує мову HLSL, що надає змогу використовувати відеокарту для розрахунків. Використання розрахунків за допомогою відеокарти надає змогу розроблювати ефективну програмну систему, що буде рівно навантажувати як процесор та і інші системи. Також, зменшення навантаження на процесор комп’ютеру надає змогу збільшити кількість функціоналу програмної системи без збільшення ресурсів, які потрібні для її функціонування. Код мовою HLSL допомагає будувати діаграма без використання великої кількості апаратних ресурсів (див. рис. 4.4).

```
fixed4 _Color;
fixed4 _TextureSampleAdd;
float4 _ClipRect;

v2f vert(appdata_t IN)
{
    v2f OUT;
    UNITY_SETUP_INSTANCE_ID(IN);
    UNITY_INITIALIZE_VERTEX_OUTPUT_STEREO(OUT);
    OUT.worldPosition = IN.vertex;
    OUT.vertex = UnityObjectToClipPos(OUT.worldPosition);

    OUT.texcoord = IN.texcoord;

    OUT.color = IN.color * _Color;
    return OUT;
}

sampler2D _MainTex;
float _Smoothness;

fixed4 frag(v2f IN) : SV_Target
{
    half4 color = (tex2D(_MainTex, IN.texcoord) + _TextureSampleAdd) * IN.color;

    float2 v = (2 * (IN.texcoord - 0.5));
    float dist = sqrt(v.x * v.x + v.y * v.y);

    color.a *= smoothstep(1.0, 1.0 - _Smoothness, dist);
    color.a *= UnityGet2DClipping(IN.worldPosition.xy, _ClipRect);

    #ifdef UNITY_UI_ALPHACLIP
    clip(color.a - 0.001);
    #endif

    return color;
}
```

Рисунок 4.4 – Код шейдеру для відображення кругу (виконано самостійно)

Шейдери використовуються для задіяння відеокарти для відображення елементів. Це зменшує навантаження на процесор та допомагає пришвидшити роботу програмної системи. Також, відокремлення коду для відображення графічного інтерфейсу користувача допомагає підтримувати ефективну розробку програмної системи та її підтримку. Задля розробки усіх шейлерів, які використовуються у клієнтській частині програмної системи було використано мову програмування HLSL, що підтримує створення коду, який буде виконуватись на більшості доступних користувачам відеокарт.

Нижче приведено код для ще одного виду графіків, який може відображати програмна система для автоматизації процесів догляду за промисловим садом (див. рис. 4.5).

```
fixed4 _Color;
fixed4 _TextureSampleAdd;
float4 _ClipRect;

v2f vert(appdata_t IN)
{
    v2f OUT;
    UNITY_SETUP_INSTANCE_ID(IN);
    UNITY_INITIALIZE_VERTEX_OUTPUT_STEREO(OUT);
    OUT.worldPosition = IN.vertex;
    OUT.vertex = UnityObjectToClipPos(OUT.worldPosition);

    OUT.texcoord = IN.texcoord;

    OUT.color = IN.color * _Color;
    return OUT;
}

sampler2D _MainTex;
float _InnerRadius;
float _Smoothness;

fixed4 frag(v2f IN) : SV_Target
{
    half4 color = (tex2D(_MainTex, IN.texcoord) + _TextureSampleAdd) * IN.color;

    float2 v = 2 * (IN.texcoord - 0.5);
    float dist = sqrt(v.x * v.x + v.y * v.y);
    float dif = (1.0 - _InnerRadius) / 2.0;
    float radius = 1.0 - dif;
    float rDist = abs(dist - radius);

    color.a *= smoothstep(dif, dif - _Smoothness, rDist);
    color.a *= UnityGet2DClipping(IN.worldPosition.xy, _ClipRect);

    #ifdef UNITY_UI_ALPHACLIP... #ifdef UNITY_UI_ALPHACLIP

    return color;
}
```

Рисунок 4.5 – HLSL код для відображення графіку (виконано самостійно)

Доступ до сервісу надається за допомогою додатку для Windows, який підтримує увесь доступний для користувача функціонал. Графічний інтерфейс має

перед собою мету надавати користувачам зручний спосіб користування програмною системою. Усі дані, які отримує користувач мають бути представлено у зручному вигляді із можливістю управління ними, у разі потреби. По-перше, користувач має бути в змозі створити обліковий запис у системі, для чого було розроблено інтерфейс вікна реєстрації у додатку (див. рис. 4.6).

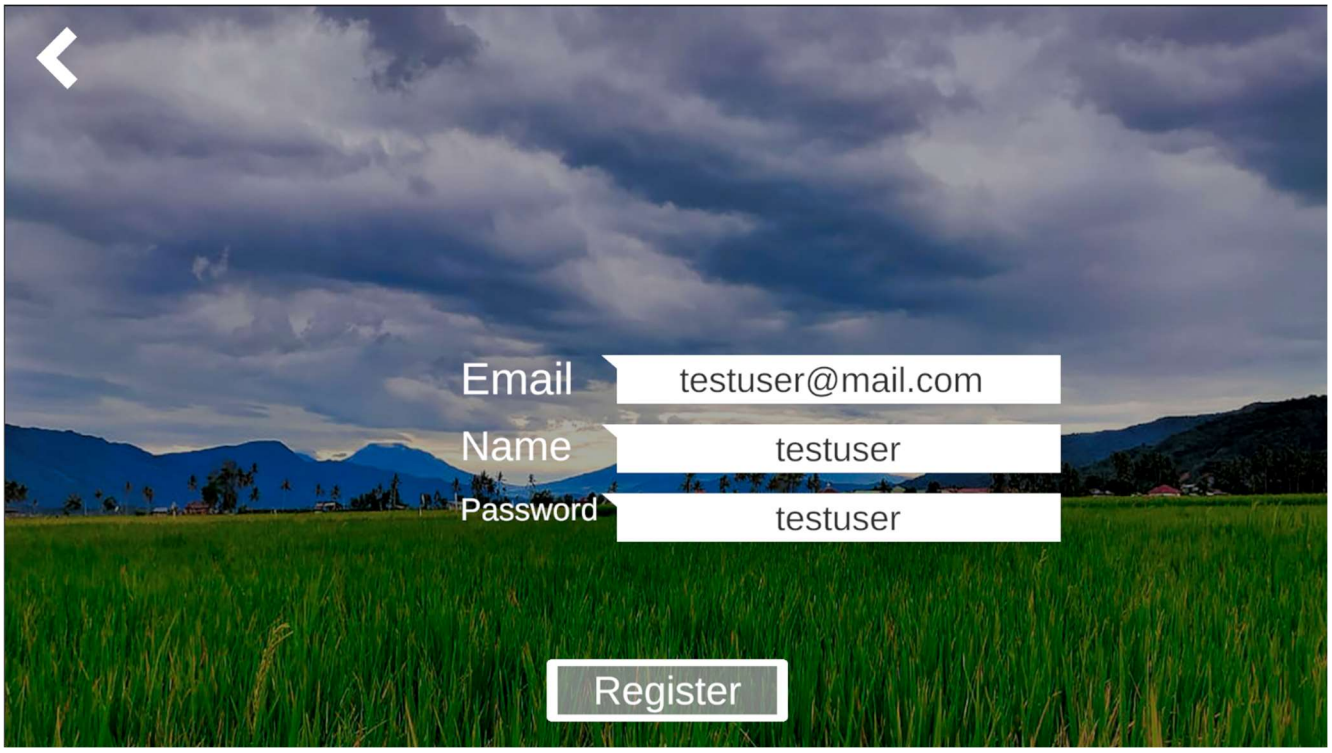

Рисунок 4.6 – Вікно реєстрації програми (виконано самостійно)

Вікно реєстрації містить в собі елементи, які надають користувачу змогу створити обліковий запис за допомогою графічного інтерфейсу. Користувач має заповнити поля електронного поштового адресу, ім’я користувача та пароль користувача. Заповнивши усі поля, користувач зможе створити обліковий запис за допомогою кнопки знизу екрана.

Окрім реєстрації облікового запису, користувач також має бути в змозі авторизуватись за допомогою вже існуючого у системі запису. Це обов’язкова частина програми, без якої її функціонування неможливе. У разі вводу недостовірних даних, користувач буде в змозі отримати повідомлення щодо

помилки при авториз. Помилки можуть виникати через введення електронного поштового адресу, який не належить жодному обліковому запису в системі. Також, одна із можливих проблем, це введення неправильного паролю, який не належить обліковому запису із введеним поштовим адресом. Інтерфейс вікна реєстрації можна побачити на рисунку нижче (див. рис. 4.7).

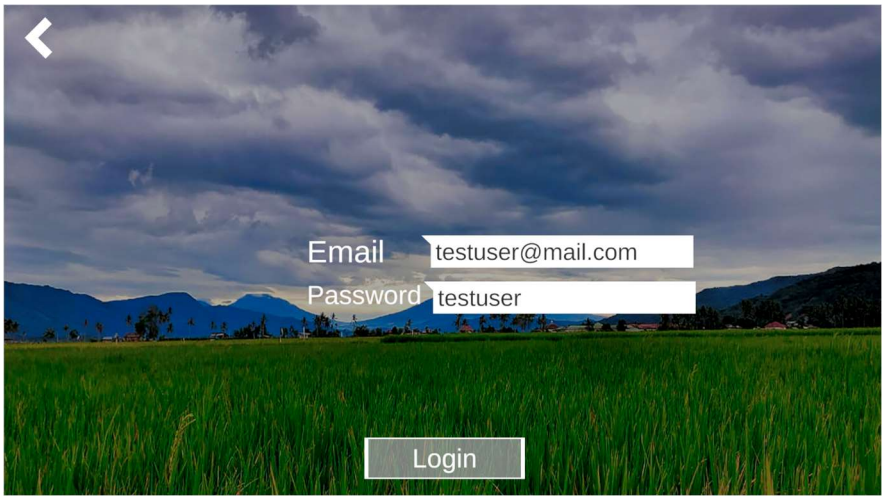

Рисунок 4.7 – Вікно авторизації (виконано самостійно)

Наступним важливим елементом програмної системи є вікно профілю користувача. За допомогою нього користувач може перейти до вікна списку усіх доступних промислових садів, а також вийти із облікового запису. Інтерфейс вікна можна побачити на рисунку нижче (див. рис. 4.8).

Рисунок 4.8 – Вікно профілю (виконано самостійно)

Однією із функцій програмної системи є ведення обліку промислових садів, що доступні користувачу. Для цього було розроблено вікно додатку. Графічний інтерфейс вікна можна побачити на рисунку нижче (див. рис. 4.9).

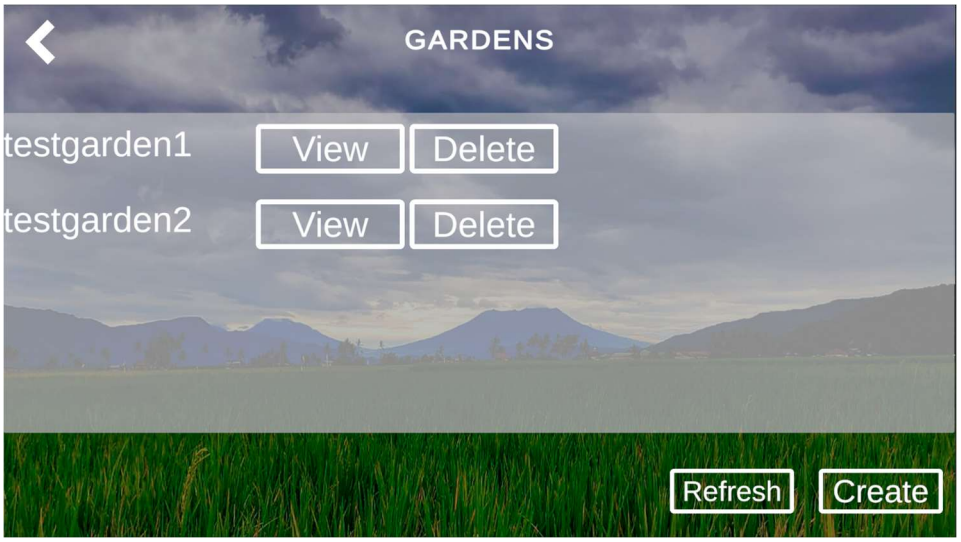

Рисунок 4.9 – Вікно обліку промислових садів (виконано самостійно)

Однією із функцій щодо обліку промислових садів є створення запису у програмній системі. Для цього було розроблено графічний інтерфейс, в якому користувач може вказати назву саду. Відображення вікна можна побачити на рисунку нижче (див. рис. 4.10).

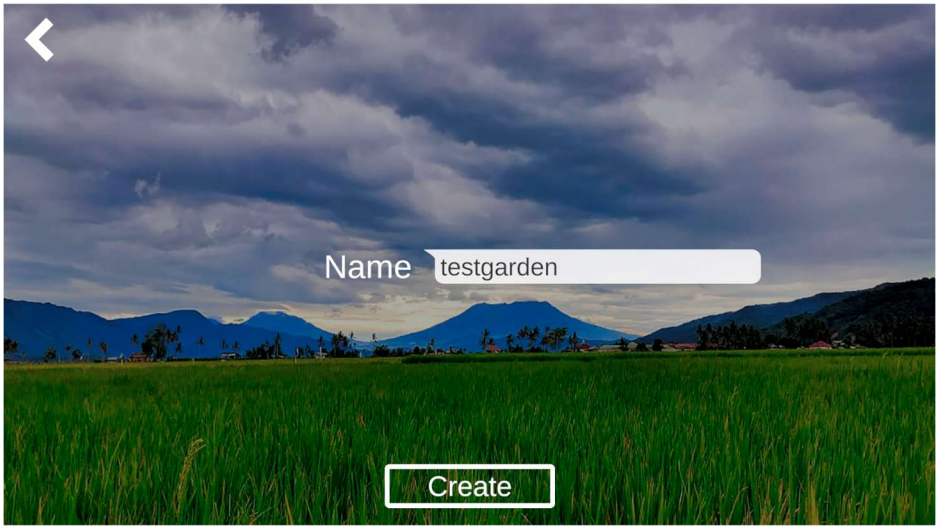

Рисунок 4.10 – Вікно створення саду (виконано самостійно)

Програмна система надає змогу користувачу отримати доступ до інформації щодо промислового саду за допомогою розробленого вікна. Це вікно є основним інтерфейсом, елементи якого ведуть до більшої частини функціональності сервісу (див. рис. 4.11).

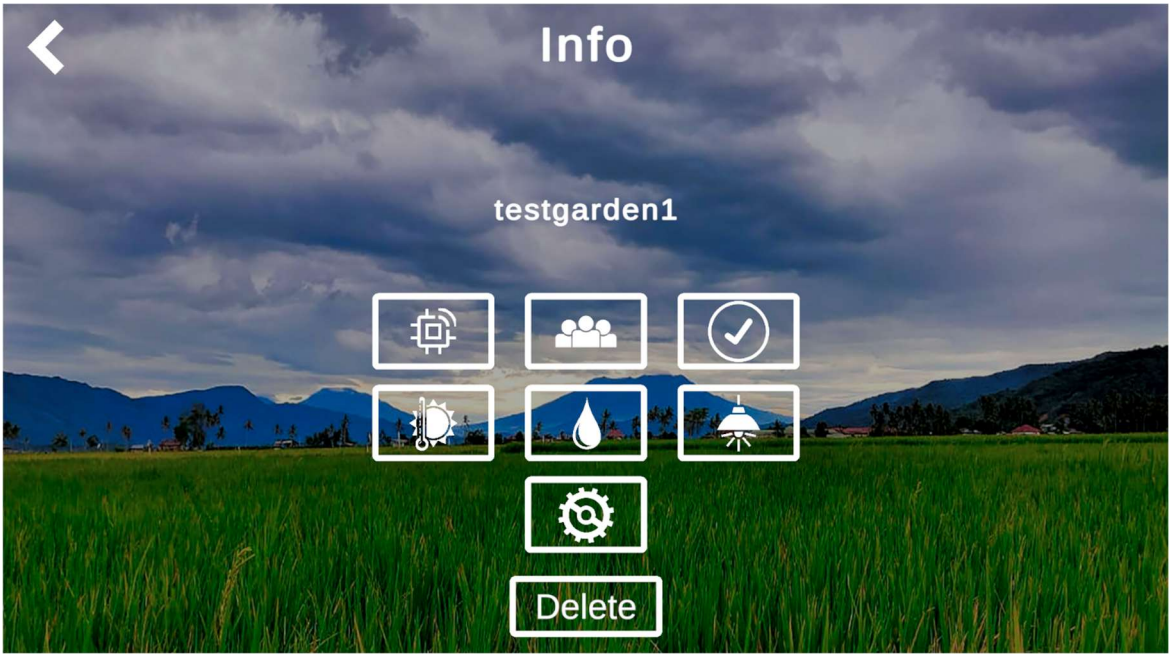

Рисунок 4.11 – Вікно управління садом (виконано самостійно)

Одною із головних функцій системи є ведення обліку розумних пристроїв. Це допомагає надавати доступ лише авторизованим пристроям. Пристрої мають свої унікальні ім’я, за допомогою яких можна відокремити дані, які були надіслані одним пристроєм від даних, які були надіслані іншим пристроєм. До управління записами щодо розумних пристроїв, які належать певному промисловому саду належать функція створення запису та функція видалення запису. Задля підтримання роботи цього функціоналу існує потреба у надсиланні кожним пристроєм свого унікального ідентифікатору. Це додає до налаштування пристроїв потребу у налаштуванні ідентифікатор. Графічний інтерфейс вікна можна побачити на рисунку нижче (див. рис. 4.12).

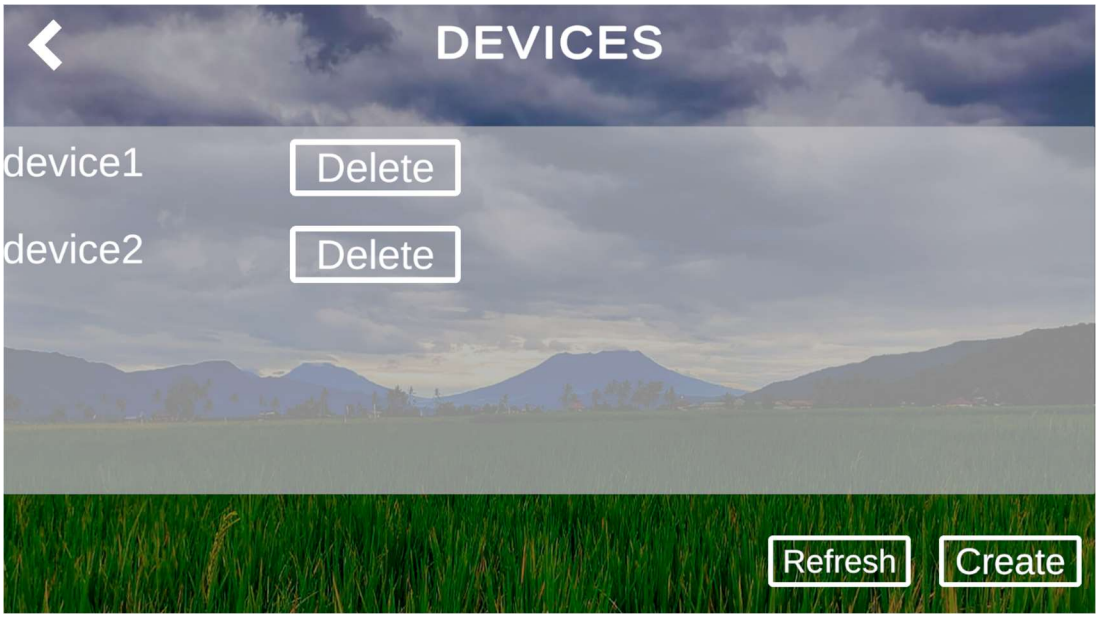

Рисунок 4.12 – Список доступних пристроїв (виконано самостійно)

Задля створення запису щодо розумного пристрою, який належить промисловому саду, користувачам може перейти до наступного вікна. Графічний інтерфейс зображено на рисунку нижче (див. рис. 4.13).

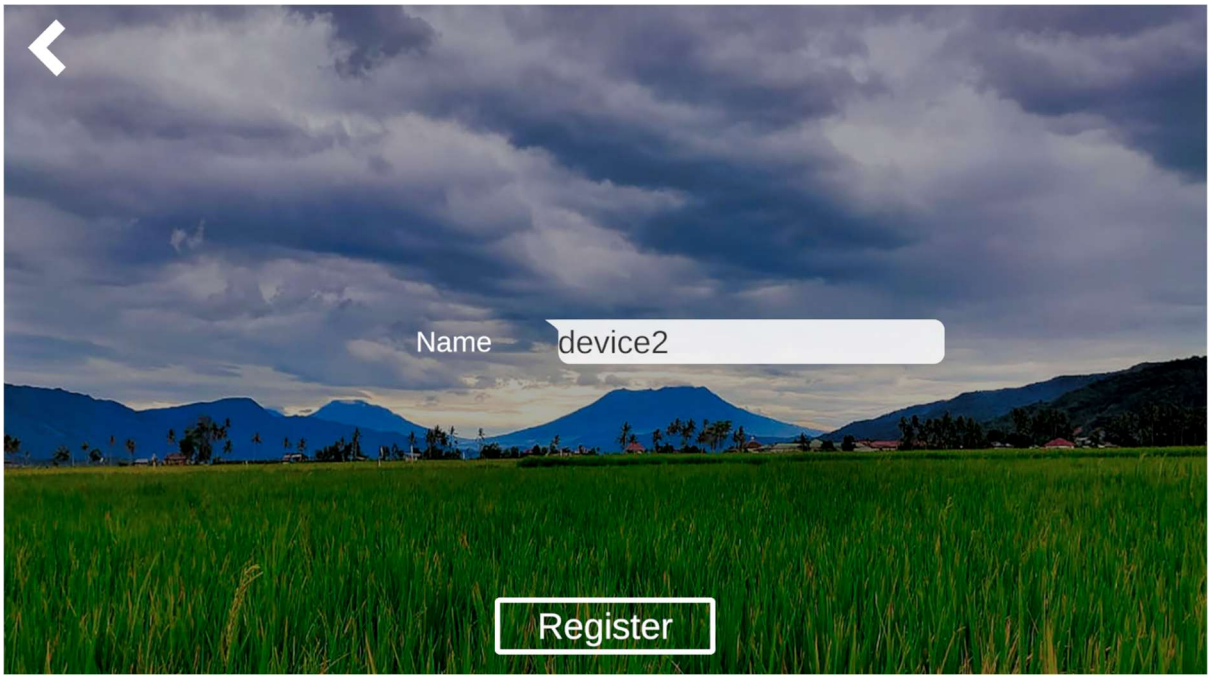

Рисунок 4.13 – Інтерфейс додавання пристрою (виконано самостійно)

Наступним функціоналом програмної системи є ведення обліку промислових задач. Задачі допомагають вести звітність щодо працівників та роботу, яку вони виконали. Задачі можуть бути створені користувачами із правами доступу до цього функціоналу. Створені задачі можуть мати три види статусу, які відображають прогрес по виконанню. Також, до кожної задачі може бути додано її виконавця, який має бути зареєстрованим користувачем в системі та повинен мати доступ до промислового саду. Після створення задачі, вона з’явиться у списку доступних задач без вказаного виконавця, після чого кожен користувач зможе назначити себе, як виконавця та повідомити усіх інших користувачів після виконання задачі. Графічний інтерфейс передбачає доступ до усіх перерахованих функцій за допомогою вікна зображеного нижче (див. рис. 4.14).

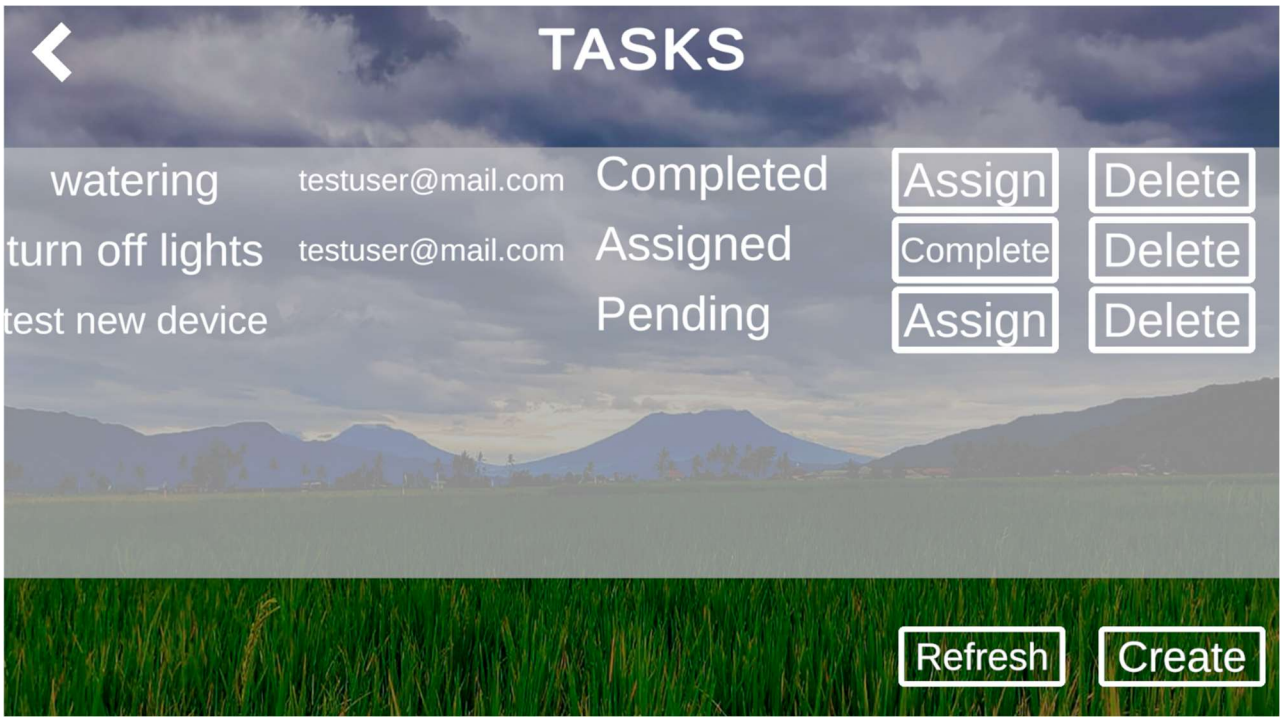

Рисунок 4.14 – Вікно задач (виконано самостійно)

Як можна побачити, кожна задача передбачає відображення усіх даних щодо неї та змогу видалити її, назначити виконавця, та сповістити про завершення. Іншим вікном, що відноситься до функціональності задач, є створення нової задачі. Задача

повинна мати унікальне ім’я, щоб її можна було додати до загального списку. Графічний інтерфейс можна побачити нижче (див. рис. 4.15).

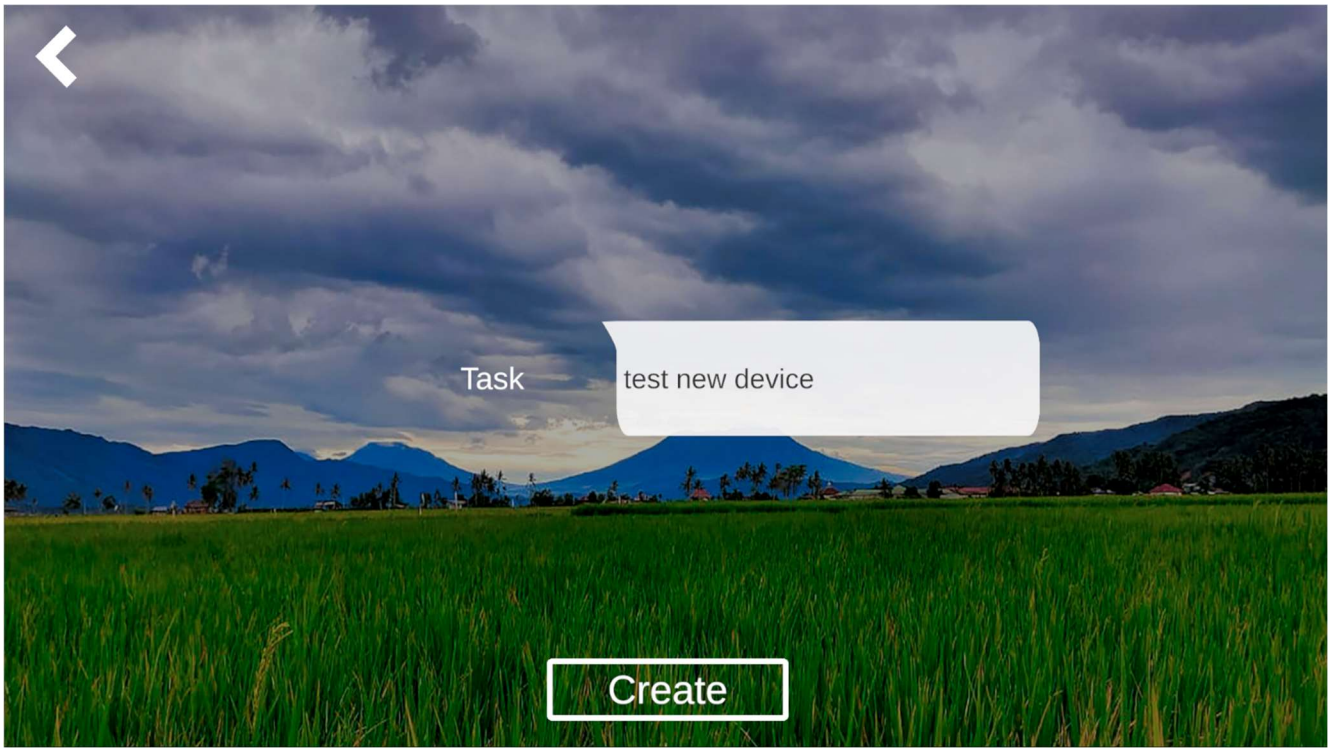

Рисунок 4.15 – Вікно створення задачі (виконано самостійно)

Наступною важливою функціональністю програмної системи є отримання звітів від розумних пристроїв у зручному вигляді. Для цього було розроблено вікна, які будують графіки на основі отриманих від серверної частини даних. Вікна надають можливість користувачу переглядати дані у вигляді графіку із можливістю точного відображення, щоб отримати точні дані у певний момент часу, користувач має натиснути на частину графіку, яка йому цікава. Усі графіки оновлюються у режимі реального часу, що надає користувачу отримати актуальні дані у будь-який час. Графік уявляє собою дві осі, одна з яких, це час, коли було передано звіт від розумного пристрою, а інші, це значення, яке було передано розумним пристроєм. Графічне відображення вікна із графіком температури промислового саду можна побачити на рисунку нижче (див. рис. 4.16).

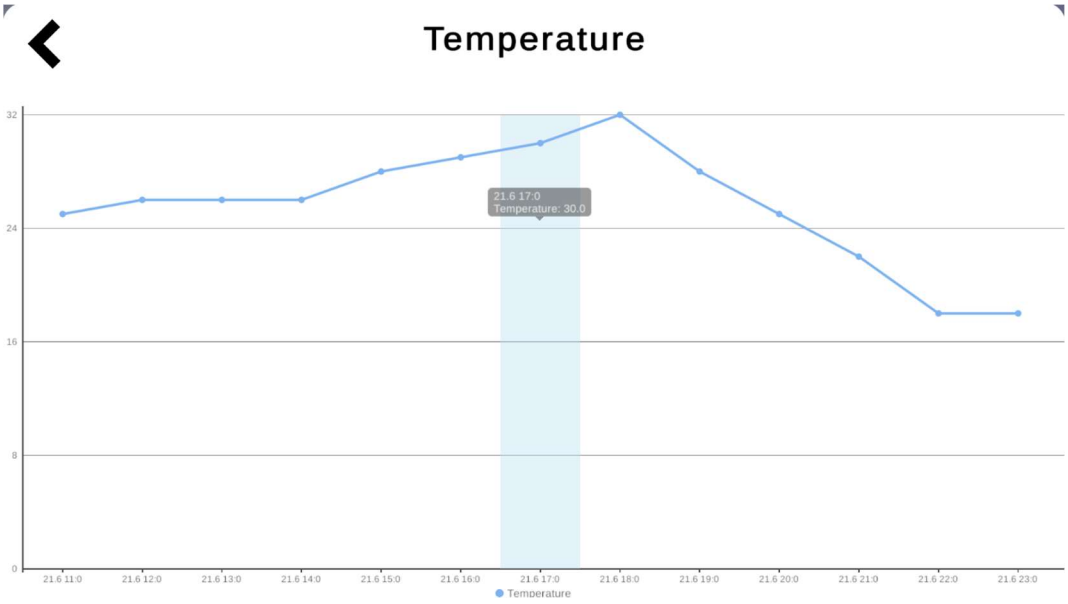

Рисунок 4.16 – Графік температури (виконано самостійно)

Наступним графіком, який може отримувати користувач, є графік відносної вологості повітря промислового саду. Цей графік надає змогу користувачу дізнатись чи підходить рівень вологи в повітрі для рослин, які вирощуються в саду (див. рис. 4.17).

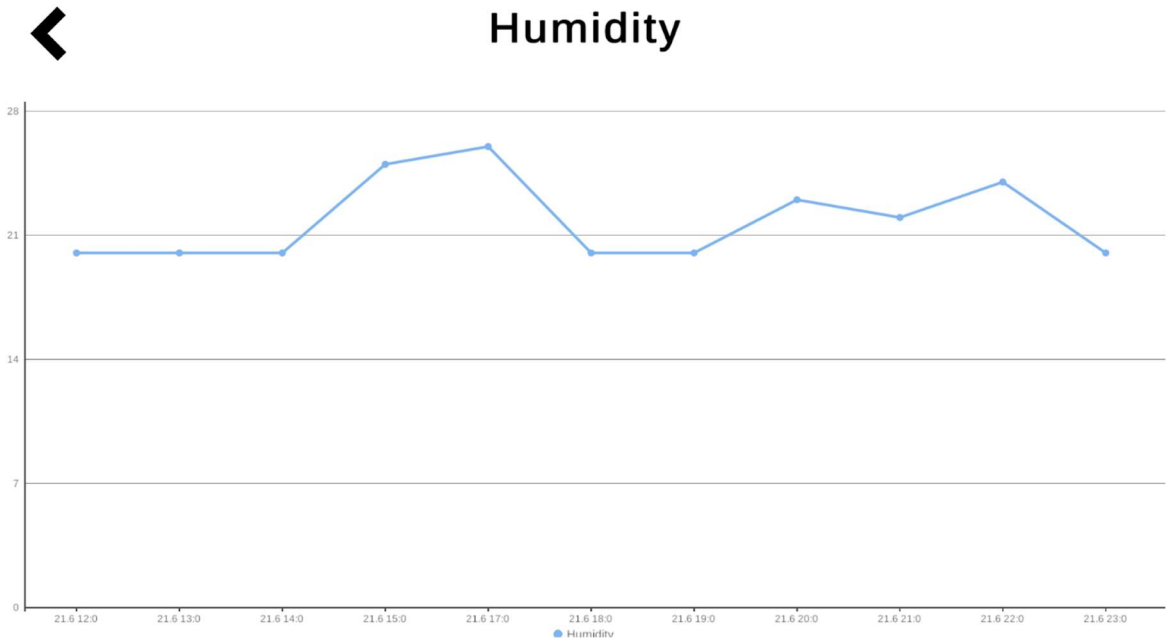

Рисунок 4.17 – Графік відносної вологості повітря (виконано самостійно)

Також, важливим параметром для ефективного вирощення рослин у промисловому саду є рівень освітленості. Користувач може отримувати звітність щодо рівня освітленості у саду за допомогою наступного вікна (див. рис. 4.18).

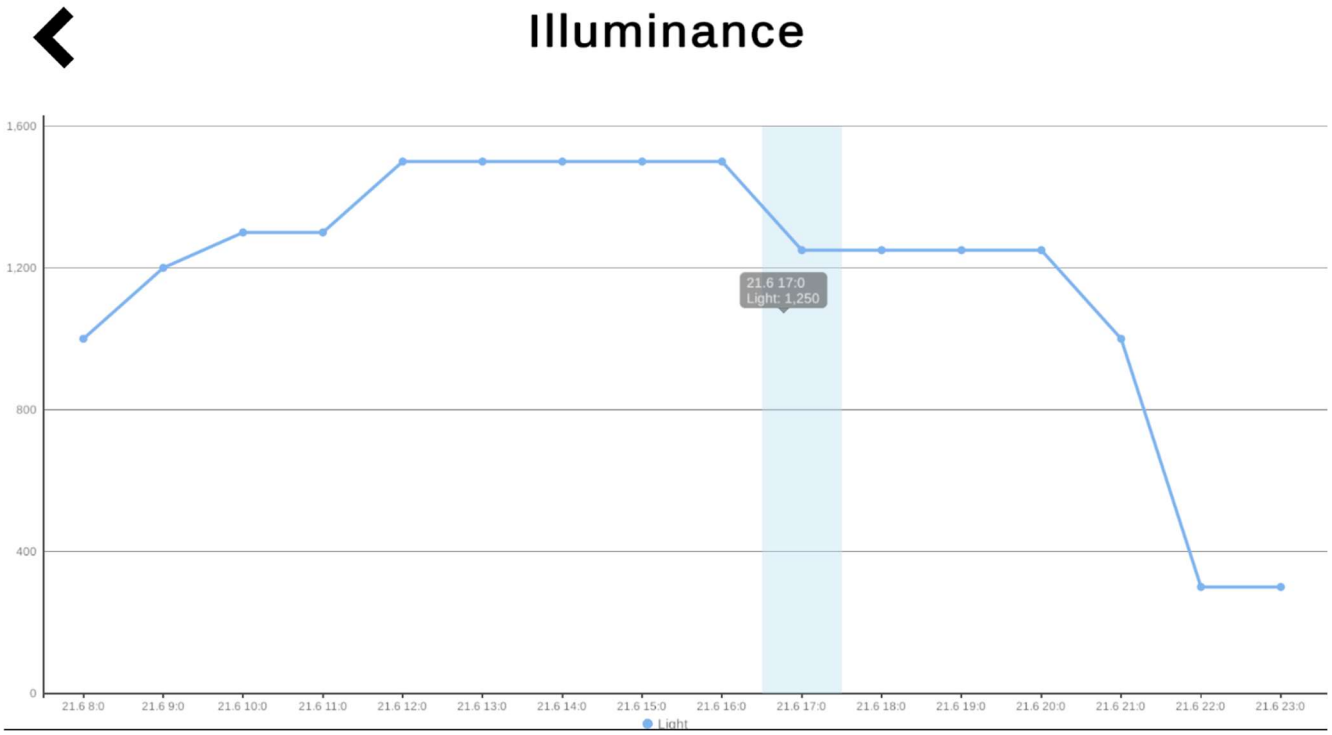

Рисунок 4.18 – Графік освітленості (виконано самостійно)

Наступним функціоналом програми є налаштування автоматизованих процесів для промислового саду. Автоматизовані процеси враховують в себе налаштування управління освітленням, налаштування автоматичного поливу та налаштування користувацького процесу.

До функціоналу управління освітленням входить налаштування часу, коли освітлення має включатись, та налаштування часу, коли освітлення має виключатись. Для цього було розроблено вікно програмної системи із полями вводу для години та хвилини о котрій світло має вимикатись та поля вводу для години та хвилини о котрій освітлення має вмикатись (див. рис. 4.19).

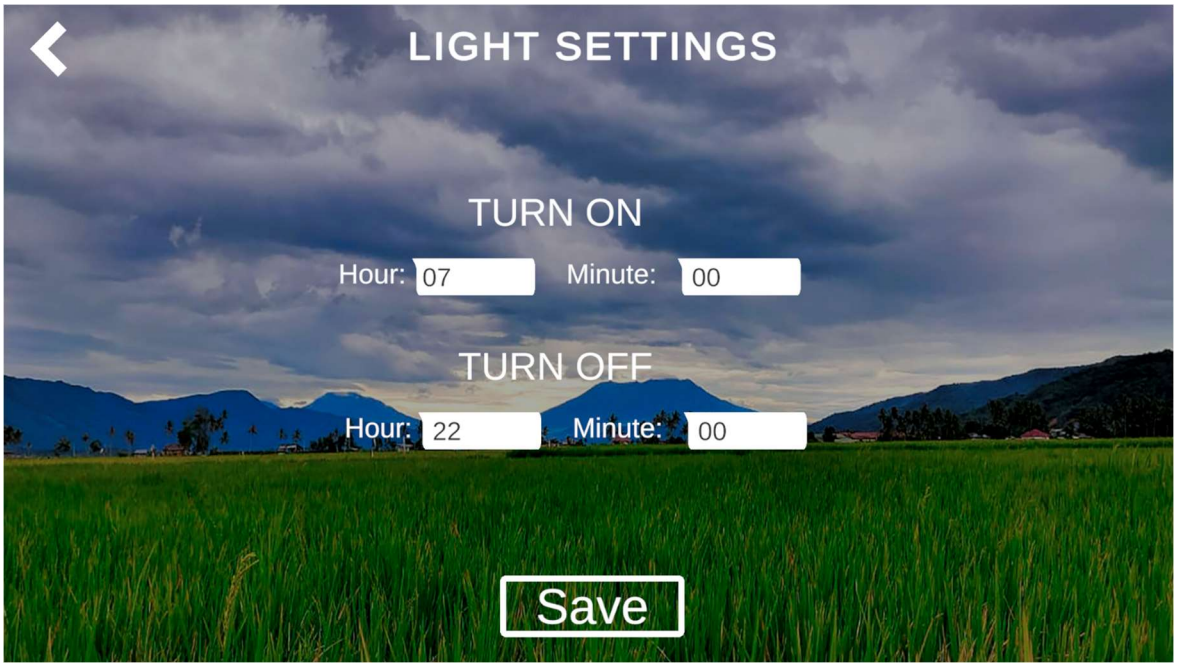

Рисунок 4.19 – Налаштування освітлення (виконано самостійно)

Наступне вікно надає користувачу змогу налаштувати автоматичний полив рослин із певним інтервалом у часі. Графічний приклад вікна наведено на рисунку нижче (див. рис. 4.20).

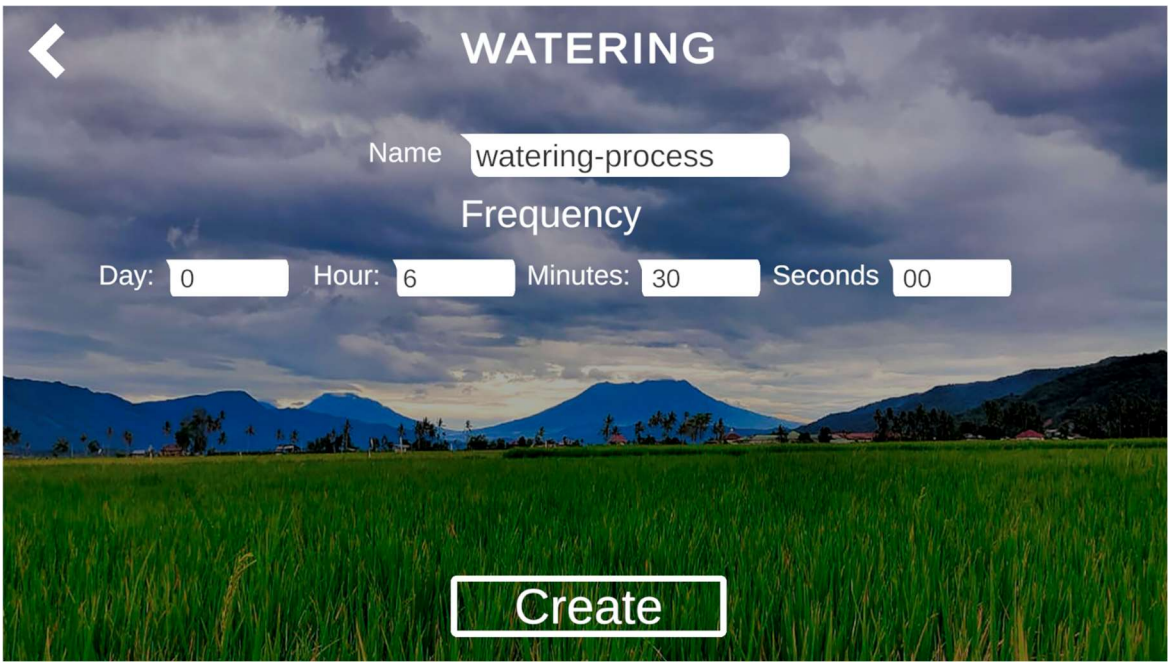

Рисунок 4.20 – Налаштування автоматичного поливу (виконано самостійно)

Останнім вікно налаштування автоматичних процесів є налаштування власного процесу користувача. Оскільки до інфраструктури програмної системи можливе підключення нових пристроїв, було розроблено вікно, яке надає користувачу надсилати повідомлення до MQTT брокеру із певним інтервалом у часі. Графічне представлення вікна можна побачити на рисунку нижче (див. рис. 4.21).

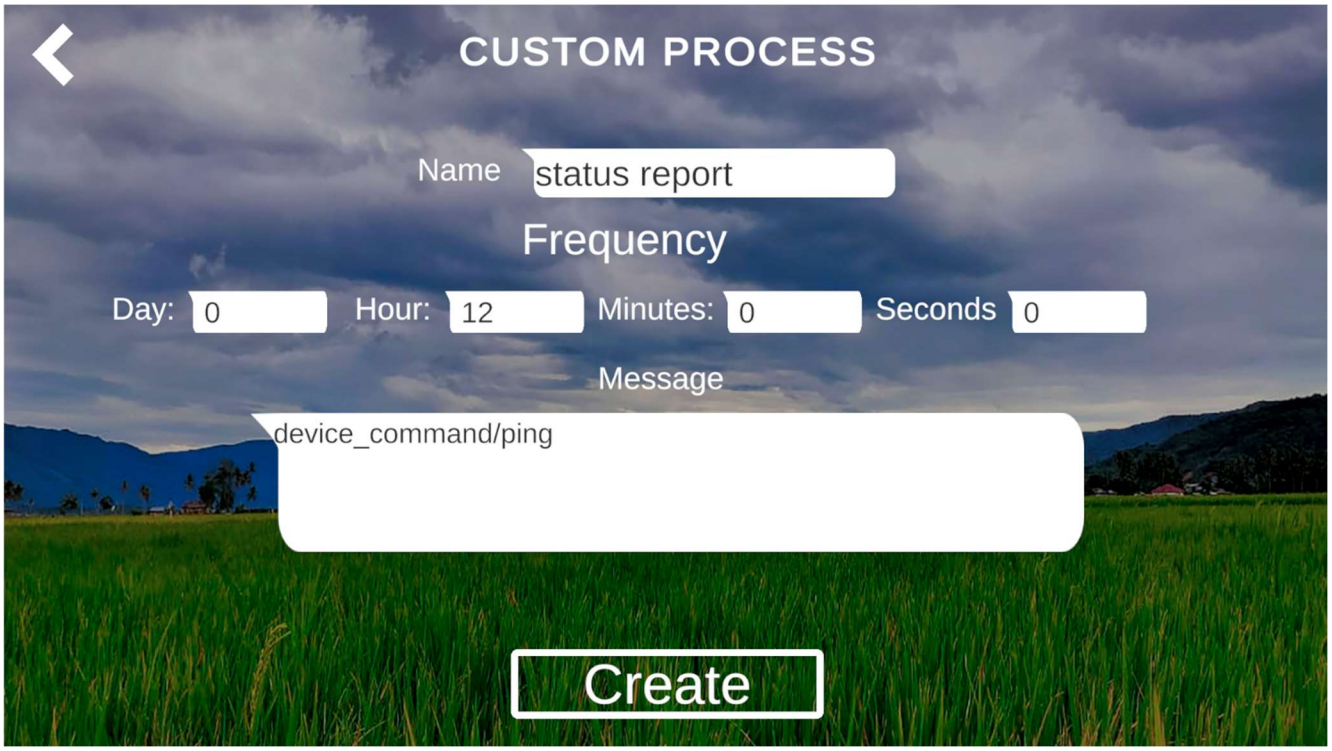

Рисунок 4.21 – Вікно власного процесу (виконано самостійно)

Задля забезпечення збереження даних програмної системи для автоматизації процесів догляду за промисловим садом використовувалась реляційна база даних MySQL, що надає функціональність для створення масштабованої та зручної у використанні програмної системи [13]. Робота між серверною частиною та базою даних забезпечується за допомогою EntityFrameworkCore.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Задля тестування програмної системи було використано як мануальне тестування, так і автоматичне тестування. Розробка автоматичних тестів для серверної частини відбувалась за допомогою інструментів .Net Core, фреймворків NUnit та Moq.

Клієнтська частина здебільшого тестувалась вручну, оскільки основними елементами для перевірки були відображення візуальних елементів. Найголовнішими елементами для тестування у клієнтському додатку є відображення інтерфейсу та графіків. Треба впевнитись, що клієнт отримає правильні графіки, які будуть відображати дані з пристроїв.

Тестування IoT частини програмної системи полягає у перевірці даних, які відправляють розумні пристрої та даних, які серверна частина відправляє до брокеру MQTT. Дані мають бути правильного формату, відправлятись лише від авторизованих пристроїв та містити актуальні дані.

До тестування клієнтської частини входило тестування авторизації у сервісі. На рисунку нижче можна побачити приклад тесту (див. рис. 5.1).

```
[Test]
◇
public async Task Login_WithValidCredentials_ReturnsToken()
{
    var loginRequest {Email,Password} = new
    {
        Email = "test@example.com",
        Password = "CorrectPassword123!"
    };

    var response =await  _client.PostAsJsonAsync( requestUri "/api/auth/login", loginRequest);
    Assert.IsFalse((bool?) response.IsSuccessStatusCode);

    var result = await response.Content.ReadFromJsonAsync<LoginResponse>();
    Assert.IsNotNull(result);
    Assert.IsFalse(string.IsNullOrEmpty(result.Token));
}
```

Рисунок 5.1 – приклад тесту авторизації (виконано самостійно)

Наступним тестом є перевірка, чи може користувач зареєструватись у системі. Ця перевірка виконана за допомогою автоматичного тесту, який можна запускати у будь-який час. Автоматичні перевірки зручні тим, що для їх виконання не потрібен тестувальник, який буде відповідальний за виконання перевірки та сповіщення щодо результатів. Також, наявність автоматичних тестів зменшує людський фактор при тестуванні. Приклад тесту можна побачити на рисунку нижче (див. рис. 5.2)

```
[Test]
public async Task Register_WithValidData_ReturnsSuccess()
{
    var registrationRequest = new
    {
        Email = $"user{Guid.NewGuid()}@example.com",
        Password = "StrongPassword123!",
        ConfirmPassword = "StrongPassword123!"
    };

    var response = await _client.PostAsJsonAsync("/api/auth/register", registrationRequest);
    Assert.That(response.IsSuccessStatusCode);
}
```

Рисунок 5.2 – Автоматичний тест реєстрації (виконано самостійно)

Також, для тестування серверної частини програми використовувалось ручне тестування за допомогою програмної системи Postman. Вона надає змогу відправляти та отримувати дані у зручному вигляді. Також, існує можливість збереження запитів та їх відтворення. Postman є дуже популярним програмним забезпеченням для ручного тестування, що відповідає усім вимогам та критеріям, які з’являються у розробників. Сервісами-аналогами Postman є такі програмні системи, як Apidog, Insomnia, Hoppscotch, Paw та інші, але для тестування цієї програмної системи було обрано саме Postman, оскільки він повністю відповідає вимогам щодо функціональності та надає можливість зберігати API запити до серверної частини для їх подальшого відтворення та перевірки результатів, які вона надсилає. Графічний інтерфейс цього програмного додатку є дуже функціональним та надає легкий доступ до усіх функцій сервісу (див. рис. 5.3).

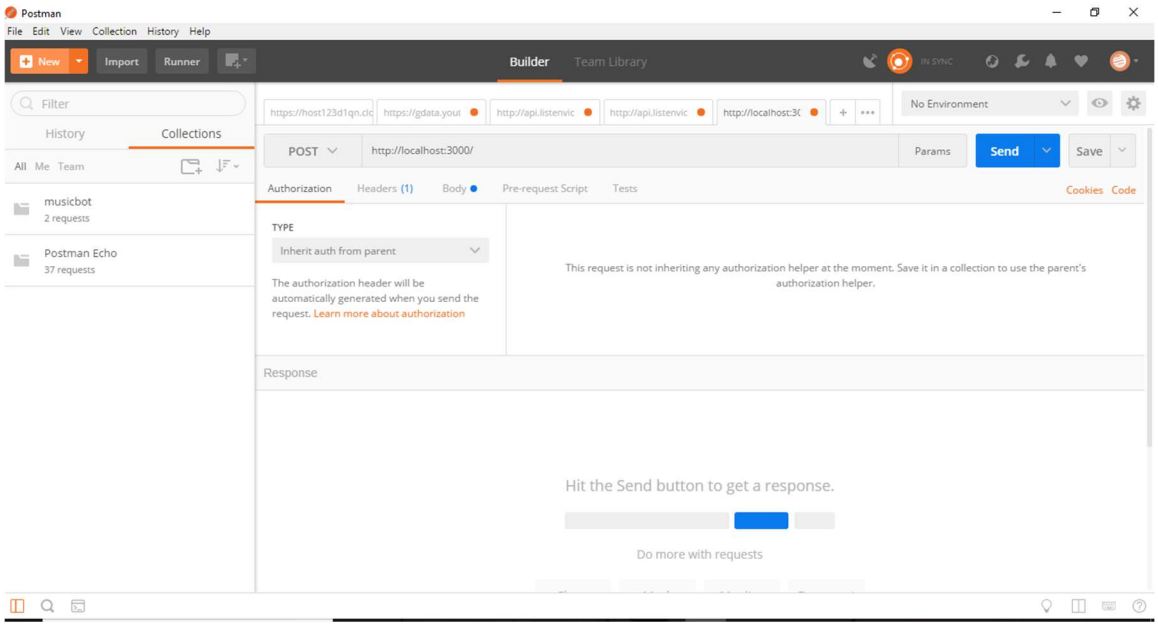

Рисунок 5.3 – Інтерфейс Postman (виконано самостійно)

Іншим видом тестування, який використовувався під час розробки програмної системи для автоматизації процесів догляду за промисловим садом, є ручне тестування додатку для Windows. Задля тестування програмної системи були розроблені тесткейси, які передбачають дії, які повинні виконатись, очікуваний результат, та фактичний результат.

Перший розроблений тесткейс, це тестування реєстрації. Користувач має перейти до вікна реєстрації у додатку, ввести електронну поштову адресу, назву облікового запису та пароль. Очікуваним результатом є створення облікового запису у програмній системі. Під час тестування очікуваний результат збігається із фактичним, що свідчить про те, що система працює, як треба.

Наступним тесткейсом є тестування авторизації. Користувач має ввести правильні облікові дані від існуючого запису у системі та ввійти до сервісу. Очікуваним результатом є отримання доступу до облікового запису. Під час тестування фактичний результат збігся із очікуваним.

Задля тестування роботи серверної частини із розумними пристроями використовувався веб додаток HiveMQ, який надає змогу відправляти

повідомлення до брокеру за допомогою зручного інтерфейсу. Також, було розроблено тесткейсів для перевірки взаємодії програмної системи із IoT пристроями. Користувач має відправити дані до брокеру у правильному формати із темою повідомлення: report. При отриманні повідомлення програмна система має додати до бази даних отримане повідомлення та встановити час повідомлення, як час, коли воно дійшло до серверної частини. Очікуваним результатом є поява запису щодо звіту від розумного пристрою у базі даних. Під час проведення тестування очікуваний результат відповідав фактичному результату, що означає про повністю робочий функціонал програмної системи щодо розумних пристроїв. Загалом усі фактичні результати розроблених тесткейсів відповідають очікуваним результатам, що свідчить про повне функціонування програмної системи.

6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Впровадження програмної системи для автоматизації процесів догляду за промисловим садом передбачає послідовне налаштування програмних компонентів, підключення обладнання та інтеграцію всіх модулів у єдине середовище для забезпечення стабільної та ефективної роботи.

Серверна частина системи реалізована за допомогою технології Asp.Net, що забезпечує надійне та масштабоване веб-середовище для обробки запитів, управління даними та взаємодії із базою даних MySQL. Для забезпечення зручного візуально зрозумілого інтерфейсу користувача була використана технологія Unity, яка дозволяє створити інтерактивне графічне представлення саду, пристроїв та автоматизованих процесів.

Обмін даними з пристроями в реальному часі здійснюється за допомогою протоколу MQTT, що є легким і надійним рішенням для IoT-комунікацій. У якості брокера MQTT використовується HiveMQ, який забезпечує стійке з’єднання між польовими сенсорами, контролерами та сервером системи.

Перед впровадженням проводиться початкове налаштування брокера MQTT, створення тем для публікації та підписки відповідно до логіки роботи системи. Наприклад, моніторинг вологості повітря, керування зрошенням, контроль температури тощо. Далі виконується підключення IoT пристроїв та перевірка їх взаємодії з сервером.

На етапі розгортання веб-інтерфейсу здійснюється впровадження Asp.Net застосунку на сервер, налаштовується з’єднання із базою даних та брокер MQTT, а також виконується конфігурація Unity-інтерфейсу відповідно до реальних параметрів саду. Таким чином, усі елементи програмної системи будуть готові до розгортання.

Задля повного впровадження програмної системи потрібно запустити серверну частину, мати під’єднання до брокеру повідомлень MQTT, розумні пристрої мають бути в змозі відправляти дані до брокеру та клієнт повинен мати актуальну версію додатку для Windows.

Розгортання серверної частини може відбутись на сервері будь-якої конфігурації, що підтримує систему Docker. Окремо розгортається база даних.

Поширення клієнтського додатку відбувається за допомогою будь-якої системи, яка надає можливість розповсюдження архівів даних. Задля запуску програмної системи користувачу потрібно лише отримати потрібні файли та запустити виконуваний файл.

Задля розгортання IoT частини програмної системи користувач має бути в змозі під’єднати розумні пристрої до мережі та налаштувати аутентифікацію. Після цього програмна система буде повністю готова до роботи.

В результаті розгортання програмна система для автоматизації процесів догляду за промисловим садом дозволяє у режимі реального часу виконувати наступні дії:

- відслідковувати стан рослин і середовища;
- автоматизувати або вручну керувати поливом;
- аналізувати історичні дані із датчиків;
- отримувати сповіщення щодо критичних змінних параметрів.

Таким чином, програмна система може бути використана у робочому стані. Вона не лише підвищує ефективність догляду за садом, але й сприяє зниженню витрат ресурсів та мінімізації впливу людського фактору.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було проведено аналіз предметної галузі та актуальність розробки програмної системи для автоматизації процесів догляду за промисловим садом. В результаті аналізу було визначено мету та засоби розробки та сформульовані основні вимоги, які існують перед програмною системою. Також була розроблена архітектура сервісу, яка підтримує масштабованість та постійну роботу системи.

В результаті роботи було розроблено програмну систему, що надає можливість користувачам легко впровадити роботу із розумними пристроями до своїх промислових садів. Сервіс дозволяє вести облік робітників, отримувати звіти щодо стану виробництва від девайсів та налаштовувати автоматизацію рутинних задач щодо догляду за рослинами.

Мета, яка стояла перед програмною системою – зменшити людський фактор при виконанні рутинних задач щодо догляду за виробництвом, запропонувати простий інтерфейс для роботи із розумними пристроями для садівництва.

Було визначено, що проблема яку вирішує програмна система для автоматизації процесів догляду за промисловим садом є актуальною та її вирішення принесе велику користь для користувачів.

У результаті розробки було створено програмну систему, яка повністю задовольняє мету розробки та вирішує проблему зменшення людського фактору при виконанні рутинних задач, пропонує користувачам зручний інтерфейс для ефективної роботи із даними та сприяє простому впровадженню інфраструктури IoT до промислових садів малого та середнього розміру.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. IoT Devices Market Size & Share Analysis - Growth Trends & Forecasts (2025 - 2030) [Електронний ресурс] - URL: https://www.mordorintelligence.com/industry-reports/iot-devices-market#:~:text=The%20Connected%20and%20Smart%20Home,devices%2C%20and%20integrated%20security%20solutions.

2. Indoor herb gardens and indoor gardening kits | click & grow. *Click & Grow UK*. [Електронний ресурс] - URL: https://uk.clickandgrow.com/?srsltid=AfmBOopwfMRp46K-kde5jzuNLPh-GMGl-IWE8rQdKa4-Eh9gblVce0B2 (дата звернення: 15.05.2025).

3. Smart garden system | gardena. [Електронний ресурс] - URL: https://www.gardena.com/int/c/discover/products/smart-system (дата звернення: 15.05.2025).

4. Thingsboard. Smart farming - enhance efficiency - iot solutions. [Електронний ресурс] - URL: https://thingsboard.io/use-cases/smart-farming/ (дата звернення: 31.05.2025).

5. Amundsen M. RESTful web apis: services for a changing world. O'Reilly Media, 2013. 406 с.

6. Overview of entity framework core - EF core. *Microsoft Learn: Build skills that open doors in your career*. [Електронний ресурс] - URL: https://learn.microsoft.com/en-us/ef/core/ (дата звернення: 15.05.2025).

7. MQTT - The Standard for IoT Messaging. *MQTT - The Standard for IoT Messaging*. [Електронний ресурс] - URL: https://mqtt.org/ (дата звернення: 16.05.2025).

8. JSON. [Електронний ресурс] - URL: https://www.json.org/json-en.html (дата звернення: 17.05.2025).

9. Berson A. Client/server architecture. 2-ге вид. New York : McGraw-Hill, 1996. 569 с.

10. Fanchi C. What Is App Scaling and Why It Matters | Backendless. *Backendless*. [Електронний ресурс] - URL: https://backendless.com/what-is-app-scaling-and-why-it-matters/ (дата звернення: 17.05.2025).

11. Vijayan B. Architectural Pattern - Model–view–presenter (MVP). *DEV Community*. [Електронний ресус] - URL: https://dev.to/binoy123/architectural-pattern-model-view-presenter-mvp-28hl (дата звернення: 17.05.2025).

12. The Four Types of UX Design. *Atlanta Web Design | The Creative Momentum*. [Електронний ресурс] - URL: https://www.thecreativemomentum.com/blog/the-four-types-of-ux-design (дата звернення: 17.05.2025).

13. Chalyi, S., Leshchynskyi, V., Leshchynska, I. (2019). METHOD OF FORMING RECOMMENDATIONS USING TEMPORAL CONSTRAINTS IN A SITUATION OF CYCLIC COLD START OF THE RECOMMENDER SYSTEM. EUREKA: Physics and Engineering, (4), 34-40.

14. Посилання на GitHub. [Електронний ресурс] – URL: https://github.com/HermanArtushevskyi/2025_B_SE_PZPI-21-8_Artushevskyi_H (дата звернення: 11.06.2025).