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 *Прохоренко Дарії Валеріївни*
(ПІБ)

академічної групи *122-19-2*
(шифр)

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

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

на тему: *Розробка комп'ютерної системи моніторингу стану мікроклімату виробничих приміщень*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*проф. Лактіонов І.С.*	95	відмінно	
розділів:				
спеціальний	*проф. Лактіонов І.С.*			
економічний	*проф. Вагонова О.Г.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

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

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

« » _____ 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-19-2* *Прохоренко Д.В.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка комп'ютерної системи моніторингу стану мікроклімату виробничих приміщень*

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

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної практики 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 _____ *Прохоренко Д.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *12,06.2023 р.*

РЕФЕРАТ

Пояснювальна записка: 62 с., 30 рис., 3 дод., 1 табл., 20 джерел.

У даній кваліфікаційній бакалаврській роботі представлено проектування та реалізацію мережі для моніторингу та контролю навколишнього середовища. Метою даної кваліфікаційної бакалаврської роботи є підвищення ефективності процесу комплексного моніторингу стану промислових приміщень за рахунок розробки комп'ютеризованої системи онлайн агрегування та обробки вимірювальних даних. Методи дослідження включають огляд літератури, системний аналіз, проектування мережі та моделювання за допомогою Cisco Packet Tracer.

Архітектура мережі складається з різних зон. Кожна зона обладнана пристроями, такими як сенсори (термостати, детектори CO2, сенсори вологості) та виконавчі пристрої (кондиціонери, обігрівачі, зволожувачі). Зв'язок всередині мережі встановлюється за допомогою комутаторів, маршрутизаторів та бездротових маршрутизаторів, що забезпечує ефективну передачу даних та управління.

Наукова новизна роботи полягає в синтезі структурної та алгоритмічної організації комп'ютерної технології моніторингу мікроклімату промислових приміщень. Він передбачає інтеграцію різноманітних сенсорів, виконавчих механізмів та комунікаційних пристроїв в єдину мережеву архітектуру, що забезпечує безперебійний обмін даними та управління в реальному часі. Такий синтез уможливлює ефективний онлайн-моніторинг комплексного мікрокліматичного стану виробничих приміщень, підвищуючи операційну ефективність і надаючи цінну інформацію для оптимізації умов праці та використання ресурсів.

Практичне значення роботи полягає в можливості моніторингу та контролю стану навколишнього середовища в різних сферах, забезпечуючи оптимальні умови праці та ефективність використання ресурсів. Представлена мережа дозволяє збирати, аналізувати та комп'ютаризовано контролювати дані в режимі реального часу, підвищуючи операційну ефективність та забезпечуючи більш безпечне та комфортне середовище. Висновки, зроблені в результаті дослідження, підкреслюють ефективність запропонованої системи інтернету речей для моніторингу та контролю навколишнього середовища.

З точки зору прогнозних припущень, очікується, що об'єкт дослідження буде продовжувати розвиватися з розвитком технологій. інтеграція з передовими аналітичними платформами, хмарними рішеннями та алгоритмами штучного інтелекту є перспективною для подальшого розширення можливостей та масштабованості системи.

Ключові слова: мережа, моніторинг навколишнього середовища, автоматизація, система управління, зв'язок, сенсори, виконавчі механізми, мережева архітектура, Cisco Packet Tracer.

ABSTRACT

Explanatory note: 63 pages.,30 fig, 3 appendix, 1 table, 20 sources.

This bachelor thesis presents the design and implementation of a network for environmental monitoring and control. The purpose of this bachelor's thesis is to improve the efficiency of the process of integrated monitoring of industrial premises by developing a computerized system for online aggregation and processing of measurement data. Research methods include literature review, system analysis, network design and modeling using Cisco Packet Tracer.

The network architecture consists of different zones. Each zone is equipped with devices such as sensors (thermostats, CO2 detectors, humidity sensors) and actuators (air conditioners, heaters, humidifiers). Communication within the network is established by means of switches, routers and wireless routers, which ensures efficient data transmission and control.

The scientific novelty of this work lies in the synthesis of structural and algorithmic organization of computer technology for monitoring the microclimate of industrial premises. It involves the integration of various sensors, actuators, and communication devices into a unified network architecture, allowing for seamless data exchange and real-time control. This synthesis enables the efficient online monitoring of the integrated microclimatic state of production facilities, enhancing operational efficiency and providing valuable insights for optimizing working conditions and resource utilization.

The practical significance of the work lies in the ability to monitor and control the state of the environment in various fields, ensuring optimal working conditions and resource efficiency. The presented network allows collecting, analyzing and automated control of data in real time, increasing operational efficiency and providing a safer and more comfortable environment. The conclusions drawn from the study emphasize the effectiveness of the proposed IoT system for environmental monitoring and control.

In terms of forward-looking assumptions, the research object is expected to continue to evolve with the advancement of technology. Integration with advanced analytical platforms, cloud solutions, and artificial intelligence algorithms is promising to further expand the capabilities and scalability of the system.

Keywords: network, environmental monitoring, automation, control system, communication, sensors, actuators, network architecture, Cisco Packet Tracer.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CO2 - Carbon dioxide

БД - База данних

IoT - Internet of Things

LAN - Local area network

OSI - The Open Systems Interconnection model

ПЗ - Програмне забезпечення

ПК - Персональний комп’ютер

HTML - HyperText Markup Language

CSS - Cascading Style Sheets

JSON - JavaScript Object Notation

ЗМІСТ

ВСТУП

У сучасному індустріальному світі, що швидко розвивається, якість продукції, безпека працівників та ефективність обладнання є вирішальними факторами успіху будь-якого бізнесу. Одним з ключових аспектів досягнення цих цілей є підтримання здорового та комфортного мікроклімату у виробничих приміщеннях. Температура, вологість, якість повітря та інші фактори можуть мати значний вплив на працездатність працівників і обладнання, а також на якість продукції, що виробляється.

Для вирішення цих завдань все більшої популярності набувають комп'ютерні системи моніторингу стану мікроклімату. Ці системи використовують датчики та інші пристрої для збору даних про різні параметри внутрішнього середовища і надають користувачам аналітику в режимі реального часу. Вони також мають можливість автоматично керувати системами опалення, вентиляції та кондиціонування повітря для підтримки оптимальних умов.

Метою даної кваліфікаційної бакалаврської роботи є підвищення ефективності процесу комплексного моніторингу стану промислових приміщень за рахунок розробки комп’ютеризованої системи онлайн агрегування та обробки вимірювальних даних. Система збиратиме дані з різних датчиків, розміщених по всьому приміщенню, оброблятиме та аналізуватиме ці дані, а також забезпечуватиме моніторинг та керування системами опалення, вентиляції та кондиціонування повітря та іншим обладнанням у режимі реального часу. Система також надаватиме інформацію та аналітику про стан навколишнього середовища в приміщенні, допомагаючи покращити здоров'я та безпеку працівників, продуктивність обладнання та якість продукції, що виробляється.

У цій роботі наведено: огляд процесу розробки, включаючи вибір датчиків, проектування архітектури системи, реалізацію програмного забезпечення та тестування системи. У ній також обговорюються потенційні переваги та обмеження системи, а також майбутні напрямки досліджень і розробок у цій галузі.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

В останні роки оцінка мікроклімату і, зокрема, теплового комфорту також використовується як один із елементів для визначення енергоефективності будівель. Оптимізація споживання енергії відповідно до термогігрометричного комфорту, створеного теплотехнічною системою та архітектурними елементами. Вивчення мікрокліматичних параметрів допомагає регулювати системи, такі як системи кондиціонування та опалення, щоб максимально знизити витрати енергії, забезпечуючи максимальний комфорт проживання. Для збереження здоров'я і благополуччя працівників важливо приділяти першочергову увагу оцінці та управлінню виробничим мікрокліматом[20]. Це включає в себе впровадження належного інженерного контролю, систем вентиляції та ергономічних практик для зменшення потенційних ризиків для здоров'я, пов'язаних з виробничим мікрокліматом[5]. Мікроклімат виробничих приміщень може створювати кілька проблем, серед яких:

- Температура: промислові приміщення часто мають великі відкриті простори, які важко ефективно обігріти або охолодити, що призводить до температурних коливань, які можуть бути незручними для працівників і потенційно впливати на роботу обладнання. Також, при низькій температурі працівники можуть відчувати дискомфорт та холод, що може призвести до зниження продуктивності та збільшення ризику травм та захворювань, таких як грип, простуда та гіпотермія. Надто низька температура також може збільшити ризик порушення моторики. Це може також призвести до зниження продуктивності, помилок на робочому місці та нездорових станів, таких як тепловий

удар та гіпертермія. На рис. 1.1 зображено можливі наслідки впливу високої температури на людину[1];

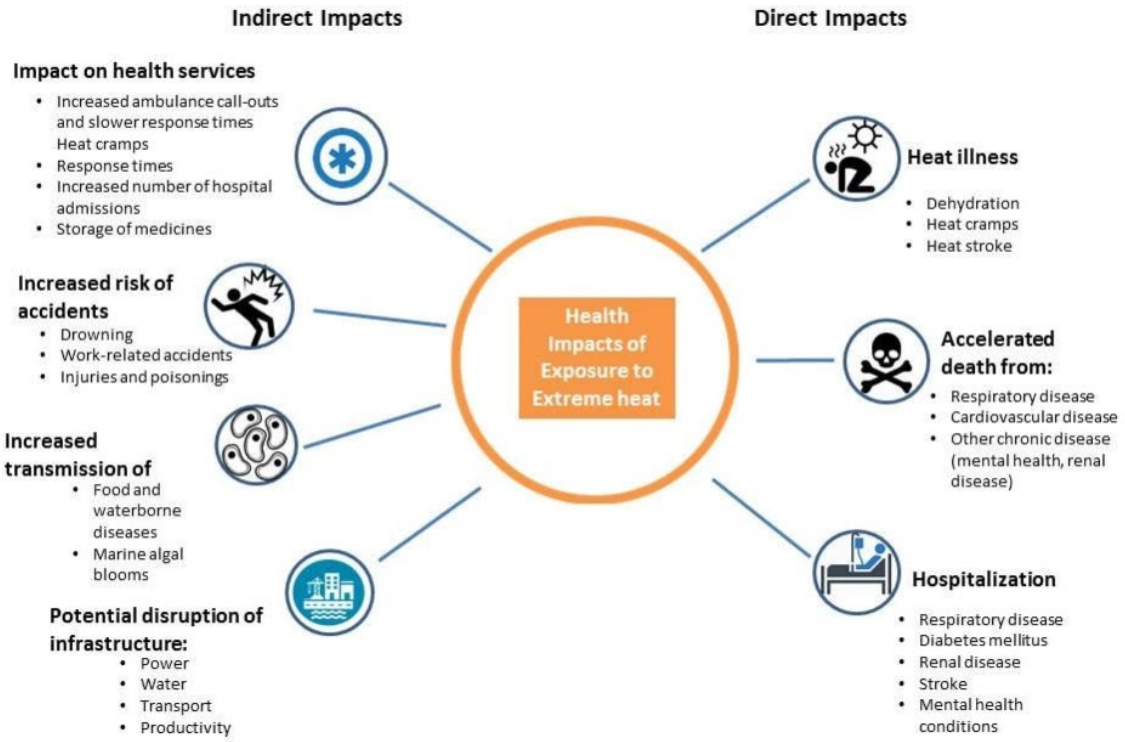

Рис. 1.1. Вплив високої температури на людину

- Вологість: певні промислові процеси можуть створювати високий рівень вологості, що може призвести до накопичення вологи, розвитку цвілі та інших проблем, які можуть вплинути на здоров’я та безпеку працівників. Зокрема, такі наслідки для здоров'я, як алергічні реакції, респіраторні захворювання, дискомфорт, втрата продуктивності є наслідком неправильного значення відносної вологості повітря. Біологічні агенти, такі як бактерії, віруси та грибки, потребують належних умов у приміщенні для свого проростання і потрапляння в повітря, а також транспортування до організму людини, як показано на рис. 1.2. Низький рівень вологості може спричинити сухість у носі, роті, очах та шкірі. Мета полягає в тому, щоб забезпечити сухість поверхонь через значення відносної вологості повітря, а також адекватну вентиляцію в приміщенні [2].

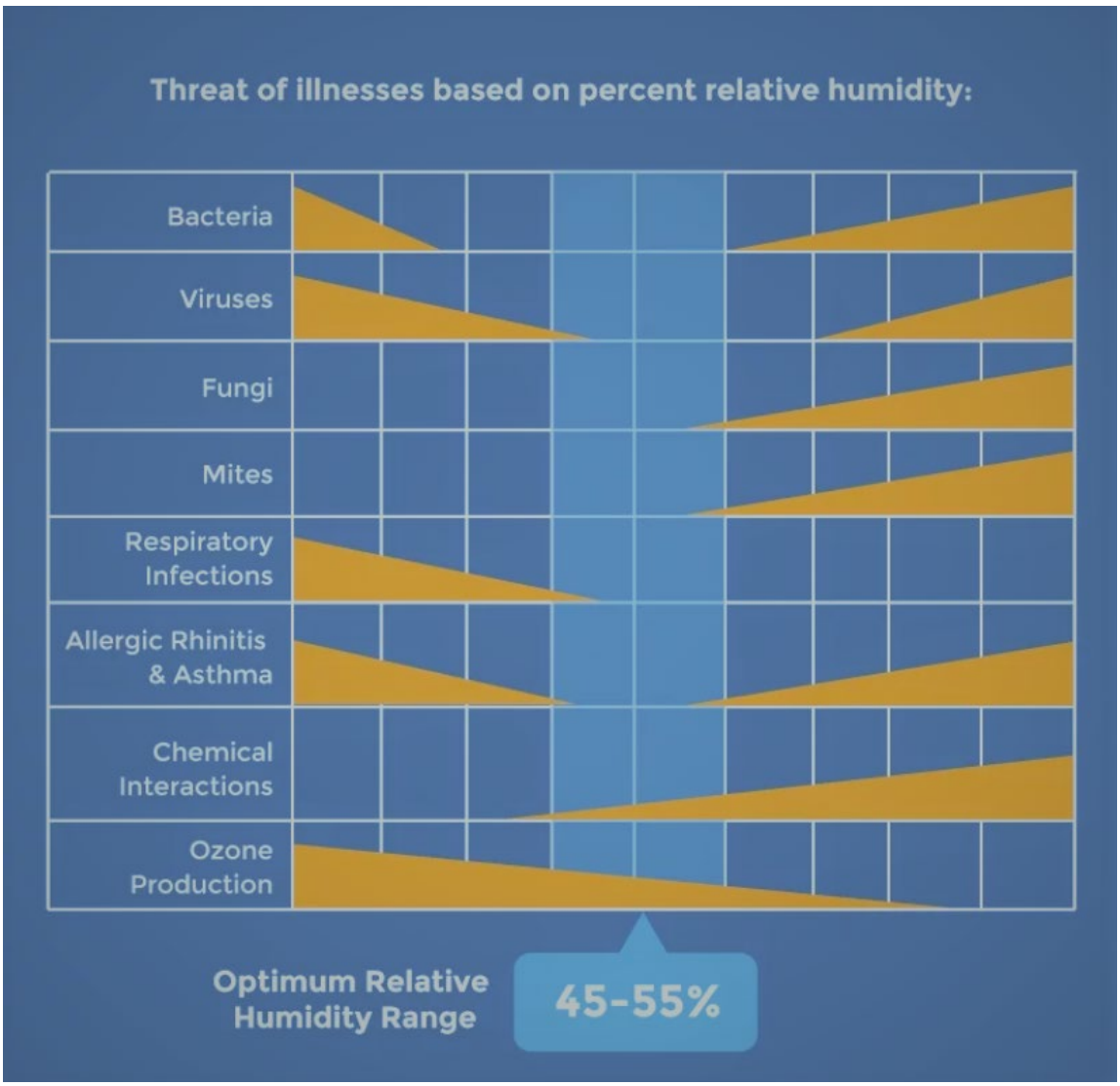

Рис. 1.2. Оптимальна вологість повітря

- Якість повітря: промислові процеси також можуть утворювати забруднювачі повітря та інші забруднювачі, які можуть впливати на якість повітря в приміщеннях, що призводить до потенційних ризиків для здоров'я працівників. Моніторинг мікроклімату промислових приміщень дозволяє виявити та вирішити ці та інші проблеми, щоб створити безпечніше, здоровіше та продуктивніше робоче середовище. Вуглекислий газ - це безбарвний газ без запаху. Він утворюється як природним шляхом, так і внаслідок людської діяльності, наприклад, спалювання бензину, вугілля, нафти та деревини. У навколишньому середовищі люди видихають CO2, що сприяє підвищенню рівня вуглекислого газу в повітрі. Рівень CO2 в приміщенні залежить від кількості присутніх людей тривалості перебування в приміщенні кількості свіжого повітря,

що надходить ззовні розміру приміщення або площі чи забруднюють повітря в приміщенні побічні продукти згоряння концентрація на вулиці. Концентрація вуглекислого газу в приміщенні може варіюватися від декількох сотень проміле до понад 1000 проміле в приміщеннях з великою кількістю людей, присутніх протягом тривалого періоду часу, і де вентиляція зовнішнього повітря обмежена (рис. 1.3). Вуглекислий газ часто вимірюється в приміщеннях, щоб швидко, але опосередковано оцінити приблизно, скільки зовнішнього повітря надходить в приміщення по відношенню до кількості мешканців. CO2 можна виміряти за допомогою відносно недорогого цифрового обладнання для моніторингу повітря в режимі реального часу. Вимірювання CO2 стало широко використовуваним скринінговим тестом якості повітря в приміщенні, оскільки його рівень можна використовувати для оцінки кількості вентиляції та загального комфорту. Належна вентиляція може обмежити накопичення цих забруднювачів. Американське товариство інженерів з опалення, охолодження та кондиціонування повітря (ASHRAE) розробило рекомендації щодо вентиляції, які повинні підтримувати комфортне середовище для більшості мешканців. Кількість свіжого повітря, що має надходити до приміщення, залежить від типу закладу та приміщення. Наприклад, для класів початкової школи ASHRAE рекомендує 15 кубічних футів на хвилину на людину зовнішнього повітря (для приміщення площею 1000 квадратних футів, в якому перебуває 35 осіб). В офісних приміщеннях ASHRAE рекомендує 17 кубічних футів на хвилину на людину (на 1000 квадратних футів, де проживає 5 осіб). Крім того, правило Міністерства праці та промисловості штату Міннесота (MNDOLI) стверджує, що "зовнішнє повітря повинно подаватися в усі робочі приміщення зі швидкістю 15 кубічних футів на хвилину на людину". Така інтенсивність вентиляції повинна утримувати концентрацію вуглекислого газу нижче 1000 проміле і створювати умови якості повітря в приміщенні, прийнятні для більшості людей [3, 4].

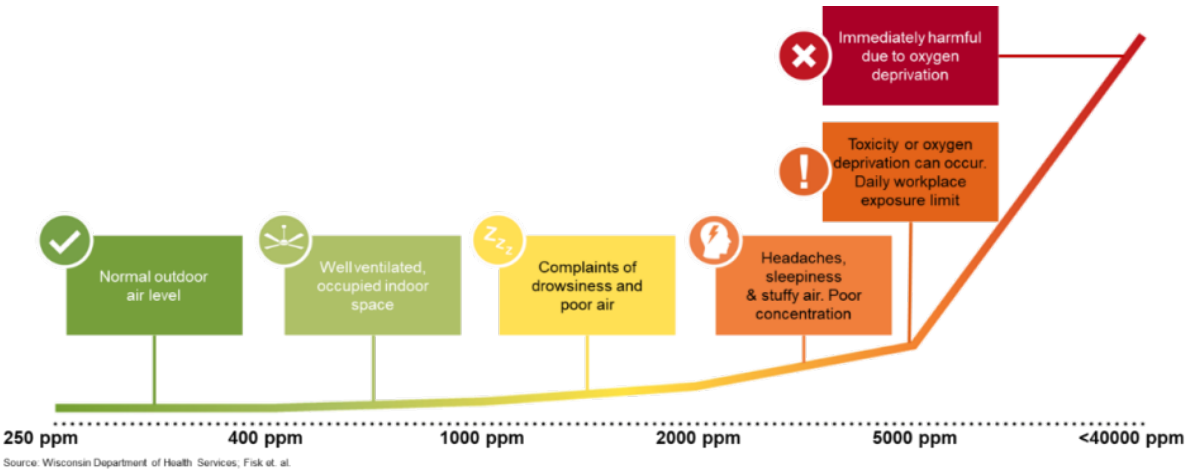

Рис. 1.3. Градація CO2 та вплив на людину

- Освітлення. Освітлення є важливим аспектом промислових приміщень, оскільки воно може впливати як на безпеку, так і на продуктивність працівників. Погане освітлення може призвести до нещасних випадків і напруження очей, а занадто яскраве освітлення може спричинити головний біль та інші проблеми зі здоров'ям.

Комп'ютерна система також може бути використана для оптимізації споживання енергії та зменшення витрат, оскільки вона може надавати дані про споживання енергії в реальному часі та пропонувати шляхи зменшення споживання. Крім того, систему можна використовувати для оптимізації продуктивності обладнання шляхом моніторингу факторів навколишнього середовища, які можуть вплинути на роботу обладнання, і надсилання попереджень або сповіщень про зміну умов.

Нижче наведено кілька прикладів комп'ютерних систем моніторингу мікроклімату виробничих приміщень:

- Onset HOBOware Pro: це програмне забезпечення для реєстрації та аналізу даних, розроблене для використання з різноманітними реєстраторами даних Onset. Систему можна використовувати для моніторингу температури, вологості, рівня вуглекислого газу та інших факторів навколишнього середовища в широкому діапазоні промислових умов, включаючи фабрики, склади та

лабораторії. Система розроблена так, щоб бути зручною для користувача та легкою в налаштуванні, і вона включає в себе ряд функцій для аналізу та візуалізації даних.

- Sensirion Industrial IoT Solution: це хмарна система, призначена для моніторингу температури, вологості та інших факторів навколишнього середовища в промислових умовах. Система включає ряд сенсорів, які можна встановити по всьому об’єкту, і дані передаються в хмару для аналізу та візуалізації. Система включає ряд функцій для сповіщення користувачів про аномальні умови навколишнього середовища, і її можна інтегрувати з іншими пристроями та платформами IoT.

- Libelium Smart Environment Pro: це бездротова сенсорна платформа, розроблена для використання в промислових умовах для моніторингу ряду факторів навколишнього середовища, включаючи температуру, вологість, світло та шум. Система включає ряд сенсорів, які можна встановити по всьому об’єкту, і дані передаються по бездротовому зв’язку на хмарну платформу для аналізу та візуалізації. Система включає ряд функцій для сповіщення користувачів про аномальні умови навколишнього середовища, і її можна інтегрувати з іншими пристроями та платформами IoT.

Загалом, це лише декілька прикладів комп’ютерних систем моніторингу мікроклімату виробничих приміщень. На ринку доступно багато різних систем, і вибір найкращої для конкретного застосування залежатиме від таких факторів, як розмір об’єкта, фактори навколишнього середовища, які потрібно контролювати, і бажаний рівень аналізу та візуалізації даних. Приклади, які було наведено, це лише деякі з багатьох комп’ютерних систем моніторингу мікроклімату промислових приміщень, які є на ринку.

1.2 Призначення розробки та галузь застосування

Метою розробки комп’ютерної системи моніторингу стану мікроклімату виробничих приміщень є забезпечення комплексне агрегування та

автоматизований інтелектуальний аналіз у реальному часі різноманітних факторів навколишнього середовища, які можуть впливати на безпеку, здоров'я та продуктивність працівників. Провівши аналіз щодо існуючих рішень системи моніторингу було сформовано наступні вимоги:

- Ефективність обладнання: деяке промислове обладнання чутливе до факторів навколишнього середовища, таких як температура та вологість. Контролюючи мікроклімат, можна оптимізувати умови для роботи обладнання та зменшити витрати на обслуговування. Загалом, мікроклімат промислових приміщень може мати значний вплив на безпеку, здоров'я та продуктивність працівників, а також на споживання енергії та вплив на навколишнє середовище. Розробивши таку систему моніторингу, можна виявити та вирішити ці проблеми, щоб створити безпечніше, здоровіше та ефективніше робоче середовище.

- Налаштування: деякі промислові об'єкти можуть мати унікальні фактори навколишнього середовища, які потребують моніторингу, і універсальної системи може бути недостатньо. У цих випадках може знадобитися налаштувати систему моніторингу мікроклімату відповідно до конкретних потреб закладу.

- Вартість: вартість системи моніторингу мікроклімату може значно відрізнятися залежно від розміру та складності об'єкта, кількості та типу необхідних сенсорів, а також необхідного рівня аналізу та візуалізації даних. Промислові об'єкти повинні враховувати вартість системи у зв'язку з перевагами, які вона надає.

- Безпека даних: промислові підприємства можуть мати конфіденційні дані, пов'язані з їхньою діяльністю, які потребують захисту. Система моніторингу мікроклімату може мати потребу включати функції, які забезпечують безпеку даних, наприклад шифрування та контроль доступу.

Загалом, це лише кілька речей, яких може не вистачати в наведених прикладах, і вони ілюструють складність проектування та впровадження ефективної системи моніторингу мікроклімату на промисловому об'єкті. Розробляючи таку систему, важливо ретельно враховувати конкретні потреби

об’єкта та співпрацювати з експертами в цій галузі, щоб переконатися, що система ефективна та відповідає всім вимогам.

Збираючи дані про температуру, вологість, якість повітря, освітлення та інші фактори, можна виявити та вирішити будь-які проблеми, які можуть виникнути на робочому місці.

Сфера застосування такої системи досить широка, оскільки її можна використовувати в будь-якому промисловому середовищі, де мікроклімат може впливати на безпеку, здоров’я та продуктивність працівників. Це стосується заводів, складів, лабораторій та інших об’єктів, які можуть піддаватися впливу факторів навколишнього середовища, таких як спека, вологість, пил або забруднювачі повітря.

1.3. Підстава для розробки

В кінці навчання, студент виконує кваліфікаційну роботу (проект). Тема роботи узгоджується з керівником проекту, випускаючою кафедрою.

Підставою для розробки кваліфікаційної роботи на тему “Розробка комп’ютерної системи моніторингу стану мікроклімату виробничих приміщень” є наказ по Національному технічному університету «Дніпровська політехніка» №350-с від 16.05.2023

1.4. Постановка завдання

Метою даної кваліфікаційної бакалаврської роботи є підвищення ефективності процесу комплексного моніторингу стану промислових приміщень за рахунок розробки комп’ютеризованої системи онлайн агрегування та обробки вимірювальних даних[6], яка дає можливість адмінітсраторам або компетентим особам відслідковувати та регулювати показники мікроклімату виробничого приміщення за допомогою графічного інтерфейсу, а також реалізація демонстраційної моделі для наглядної симуляції роботи даної системи.

Структуру об'єктів інформаційної системи в змодельованій мережі можна описати наступним чином:

- Сенсори: Це фізичні пристрої, розгорнуті в різних областях мережі для збору даних з навколишнього середовища. Приклади включають сенсори температури, сенсори вологості, детектори CO2 та інші відповідні сенсори.

- Виконавчі механізми - це пристрої, які взаємодіють з фізичним середовищем на основі інструкцій, отриманих від системи. Прикладами є кондиціонери, обігрівачі та зволожувачі повітря, якими можна керувати для регулювання умов навколишнього середовища.

Вихідна інформація з системи включає:

- Дані з сенсорів у реальному часі: Показання температури, рівня вологості, концентрації CO2 та інших параметрів навколишнього середовища, зібрані сенсорами.

- Інструкції з управління: Команди, що надсилаються на виконавчі механізми для регулювання умов навколишнього середовища на основі даних сенсорів і попередньо визначених порогових значень.

- Стан системи та сповіщення: Індикатори та сповіщення про стан системи, в тому числі про будь-які аномальні умови або події, які потребують уваги.

Збір і передача вхідної інформації в мережі повинні відповідати наступним вимогам:

- Надійний збір даних: Забезпечення точного і послідовного збору даних з сенсорів.

- Ефективна передача даних: Використання відповідних протоколів і мережевих конфігурацій для безперебійної передачі даних між пристроями.

Умови завершення автоматизованого вирішення завдань залежать від конкретних сценаріїв і можуть бути визначені на основі системних вимог. Прикладами можуть бути досягнення заздалегідь визначених умов навколишнього середовища, втручання користувача або завершення певного періоду часу.

Зв'язок цієї системи з іншими завданнями або системами може включати інтеграцію з більшими корпоративними системами, платформами аналізу даних або зовнішніми системами моніторингу та контролю. Це забезпечує обмін

даними, координацію та інтероперабельність з іншими компонентами загальної інфраструктури.

Розподіл функцій між персоналом і технічними засобами залежить від завдань системи та експлуатаційних вимог. Персонал може відповідати за моніторинг системи, аналіз даних, конфігурацію налаштувань і прийняття рішень на основі інформації, наданої мережею. Технічні засоби, такі як вбудовані в систему алгоритми автоматизації та логіка управління, можуть виконувати обробку даних в реальному часі, управління виконавчими механізмами і моніторинг стану системи.

У різних ситуаціях вирішення завдань системи розподіл функцій може змінюватися. Наприклад, під час рутинних операцій персонал може зосередитися на моніторингу та аналізі, а технічні засоби - на обробці даних та управлінні. У разі збоїв у роботі системи або аварійних ситуацій персонал може брати активнішу участь у прийнятті рішень і ручному втручанні, доповнюючи автоматизовані функції системи.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Запропонована система має наступні функціональні характеристики (рис. 1.4):

- Сенсори збирають дані мікроклімату та передають їх до сховища, які надалі передаються до клієнтської машини.

- Виконавчі механізми мають отримувати дані від клієнта і транслювати ці вхідні дані у дію. Наприклад: увімкнути обігрівач.

- Авторизація користувача: лише авторизований користувач має доступ до системи та можливість змінити налаштування.

- Користувач має змогу віддалено управляти деякими механізмаим системи.

- Сповіщення користувача за електронною адресою.

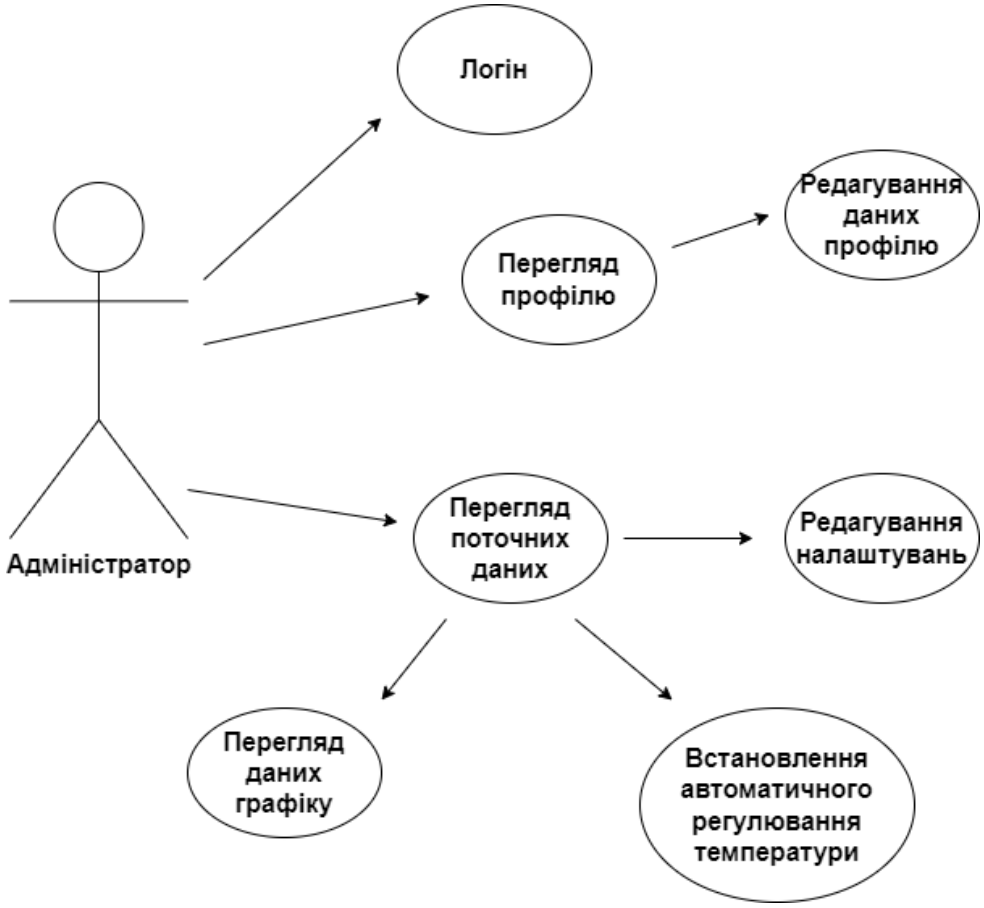

Рис. 1.4. Use Case Diagram

1.5.2. Вимоги до інформаційної безпеки

Дана система, з огляду на свої особливості та складність, не підходить для широкого кола користувачів. Щоб забезпечити належний рівень безпеки та контролю, доступ до системи обмежений і має право на нього лише обмежена група адміністраторів. Вони відповідають за налагодження та оновлення налаштувань. Один з важливих заходів, що були впроваджені з метою забезпечення безпеки, - це впровадження системи авторизації користувача. Це означає, що перед отриманням доступу до системи користувач повинен пройти процедуру авторизації, що забезпечує ідентифікацію та перевірку його прав доступу. Такий підхід дозволяє підтримувати безпеку системи та контролювати доступ до неї, зменшуючи ризик несанкціонованого використання чи недостатньо кваліфікованого впливу на систему.

1.5.3. Вимоги до складу та параметрів технічних засобів

Клієнтська машина (смартфон, ноутбук, комп’ютер, планшет). Будь-який пристрій, що може відображати веб-застосунок для клієнта.

Air Cooler, Heater - Впливають на температуру в приміщенні (зменшують або збільшують).

Thermostat - пристрій, який автоматично регулює температуру або активує пристрій, коли температура досягає певного значення.

Humidifier - це пристрій або прилад, призначений для підвищення вмісту вологи або вологості повітря в певній зоні або приміщенні. Це досягається шляхом виділення водяної пари або пари в навколишнє середовище, тим самим підвищуючи загальний рівень вологості. Зволожувачі повітря бувають різних типів, включаючи випарні, ультразвукові та парові моделі, кожен з яких використовує різні механізми для створення та розсіювання вологи.

Temperature monitor - це пристрій, що використовується для вимірювання та відображення температури певного середовища або об'єкта. Він призначений для надання точних показань температури в режимі реального часу з метою моніторингу. Монітори температури можуть варіюватися від простих ручних термометрів до більш досконалих цифрових сенсорів або мережевих систем.

Humidity monitor - Монітор вологості, також відомий як гігрометр, - це пристрій або прилад, що використовується для вимірювання та відображення рівня вологості в повітрі або певному середовищі. Він допомагає визначити кількість вологи, присутньої в повітрі, шляхом вимірювання відносної вологості.

Детектор вуглекислого газу або CO2 монітор - це пристрій, який використовується для вимірювання та моніторингу концентрації вуглекислого газу в повітрі. Вуглекислий газ - це газ без кольору і запаху, який природним чином присутній в атмосфері, але високий рівень CO2 може бути шкідливим для здоров'я людини і вказувати на потенційні екологічні проблеми.

Photo sensor - Фотосенсор, також відомий як фотосенсор або фотоелемент, - це пристрій, який виявляє наявність або відсутність світла і перетворює його в електричний сигнал. Він широко використовується в різних сферах застосування для вимірювання та контролю освітленості. Фотосенсори працюють за принципом фотоелектричного ефекту, коли падаюче світло викликає вивільнення електронів в матеріалі сенсора, що призводить до зміни електричної провідності або напруги.

Router - це мережевий пристрій, який пересилає пакети даних між комп'ютерними мережами. Він діє як центральна точка з'єднання і керує потоком даних між кількома пристроями в мережі або між різними мережами, наприклад, Інтернетом.

Switch - це мережевий пристрій, який з'єднує декілька пристроїв у локальній мережі (LAN) і полегшує обмін даними між ними. Він працює на канальному рівні (Layer 2) моделі OSI і використовує MAC-адреси для пересилання пакетів даних. Основна функція комутатора - створення сегмента мережі або локальної мережі шляхом надання декількох портів для підключення пристроїв, таких як комп'ютери, сервери, принтери та інше мережеве обладнання. Коли пристрій надсилає дані іншому пристрою в тій же мережі, комутатор використовує MAC-адресу призначення в пакетах даних, щоб визначити відповідний порт для перенаправлення пакетів, фактично встановлюючи пряме з'єднання між відправником і одержувачем.

Server - це комп'ютер або система, яка надає послуги, ресурси або функціональність іншим комп'ютерам або пристроям, відомим як клієнти, в мережі. Сервери призначені для виконання конкретних завдань і обслуговування запитів клієнтів, надаючи їм доступ до різних ресурсів і послуг.

1.5.4. Вимоги до інформаційної та програмної сумісності

Система складатиметься з апаратних та програмних компонентів. Апаратна частина буде змодельована за допомогою ПЗ Cisco Packet Tracer - це інструмент

візуальної симуляції, який дозволяє створити топологію системи та протестувати її у змодельованому середовищі. Шляхом з'єднання сенсорів утворюється мережа, а інформацію про параметри навколишнього середовища можна відстежувати за допомогою ПК. Кожен із девайсів, що збирає та обробляє дані реалізований на мові Python.

Для функціонування клієнтської частини - веб-інтерфейсу система має бути реалізована з використанням HTML, CSS, JavaScript для реалізації веб-ресурсу. Для належного функціонування веб-орієнтованої підсистеми на обчислювальній машині необхідно мати програмне забезпечення, що відповідає таким вимогам:

- Підтримка операційних систем, таких як Unix, Linux або Microsoft Windows XP/7/8/10;
- Наявність сумісного веб-браузера, такого як Microsoft Internet Explorer, Mozilla Firefox, Opera або Google Chrome;

Серверна частина реаліізована на мові Python, з використанням фреймоврку Flask;

РОЗДІЛ 2

ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1 Функціональне призначення програми

Проаналізувавши наявні приклади, функціональне призначення даної системи полягає у створенні комплексної системи із збором та аналізом даних. Розглянемо більш детально функціональне призначення:

- Збір та аналіз даних. Одними з головних призначень системи є збір даних із сенсорів про мікроклімат у приміщенні та подальший аналіз отриманих даних. Таким чином, користувач може здійснювати моніторинг в режимі реального часу і у разці виникнення відхилень вжити відповідних заходів.
- Автоматизоване управління. Користувачу надається можливість встановлювати порогові значення показників мікроклімату, відхилення від яких, сигналізуватиметься сенсорами.
- Сповіщення. Користувач має змогу отримувати регулярні сповіщення на електронну пошту для перегляду поточного стану показників у приміщенні.
- Графічний інтерфейс та візуалізація даних. Перегляд показників здійснюється за допомогою веб-додатку. Дані відображаються у реальному часі, є можливість зручного оновлення порогових значень та візуалізація даних за певний проміжок часу у вигляді таблиці.

2.2 Опис застосованих математичних методів

При написані даної кваліфікаційної бакалаврської роботи використовувалися наступні формули для обчислень:

- визначення поточного часу (формула 2.1);
- використання електроенергії за певний проміжок часу (формула 2.2);
- витарти на електроенергію (формула 2.3);

$$CT = \Delta t + ST, \tag{2.1}$$

де CT - поточний час,

ST - остання записана дата,

Δt - зміна часу для запису, у даній системі цей час становить 3 години.

$$EU = (CL/100) \cdot LU \cdot t, \tag{2.2}$$

де EU - використана електроенергія,

CL- поточна освітленність приміщення у відсотках,

t - час використання електроенергії.

$$EC = EU \cdot ECP, \tag{2.3}$$

де EU - використана електроенергія,

EC- витрати за використану електроенергію,

ECP- вартість використання електроенергії за годину.

2.3 Опис використаних технологій та мов програмування

При написанні даної кваліфікаційної бакалаврської роботи застосовується мова програмування Python - це проста та зрозуміла мова програмування, програмний код якої легко підтримувати і надалі. Відповідно, до мови програмування легко інтегрується Flask був обраний у якості фреймворку для бекенду, який надає широкий спектр бібліотек для обробки та аналізу даних, які можна використовувати в системі[18]. Також, цей фреймворк є придатним для створення масштабованих веб-додатків.

Для розробки клієнтської частини було застосовано такі фронтенд технології як:

HTML, CSS, JavaScript - за допомогою цих технологій стврюється структура веб-інтерфейсу, додаються стилі і логіка передачі та отримання даних.

2.4 Опис структури програми та алгоритмів її функціонування

Узагальнена схема роботи системи моніторингу виглядає наступним чином (рис. 2.1):

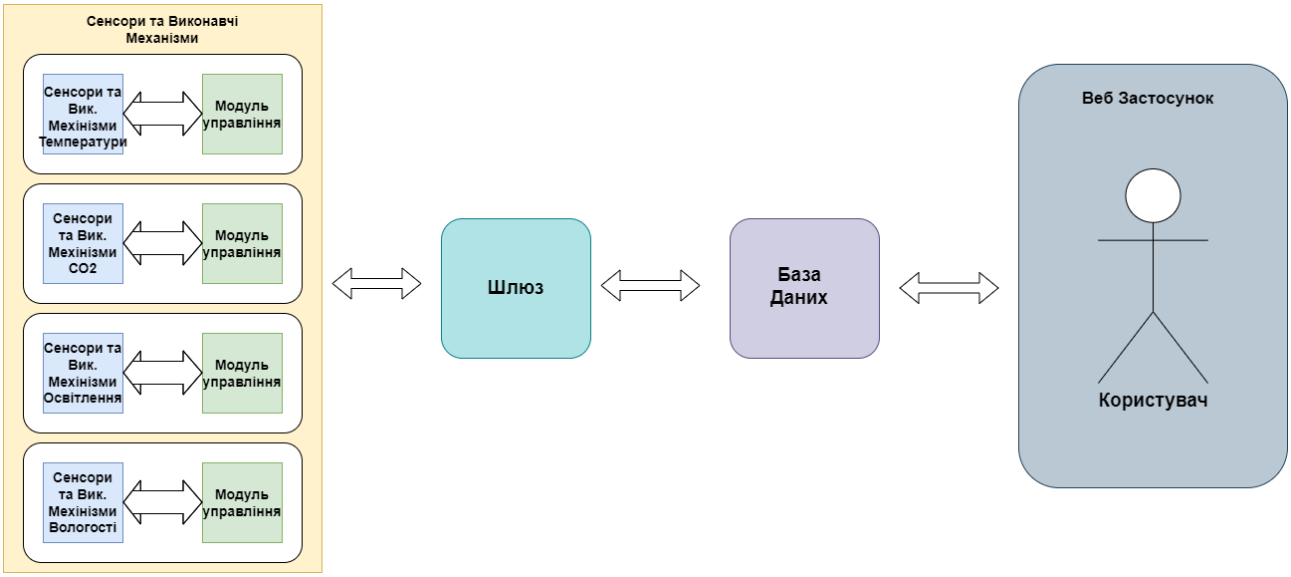

Рис. 2.1. Схема роботи системи

Далі описано більш детальний алгоритм роботи системи:

- Інтеграція сенсорів та зв’язок. Для належної роботи системи у приміщенні мають бути встановленні сенсори, які збиратимуть дані мікроклімату та виконавчі механізми, які працюють відповідно до налаштувань і отриманих даних. Усі встановлені пристрої утворюють мережу. Мережа складається з різних компонентів, включаючи сенсори, виконавчі механізми, комутатори, маршрутизатори, ПК і сервер, які пов'язані між собою. Комутатори забезпечують ефективну та надійну передачу даних між сенсорами, виконавчими механізмами та іншими пристроями мережі. Маршрутизатори відіграють вирішальну роль у з'єднанні різних сегментів мережі, забезпечуючи

безперебійний зв'язок та обмін даними між ними. Вони забезпечують потік даних між зонами.

- Аналіх даних. Отримані дані проходять алгоритми обробки та зберігаються у БД для подальшого використання.

- Управління. На основі проаналізованих даних система реалізує механізми управління для регулювання параметрів мікроклімату. Це включає управління системами опалення, вентиляції, охолоджувачами, обігрівачами, зволожувачами для підтримки оптимальних умов.

- Інтерфейс користувача: Система надає зручний інтерфейс у вигляді веб-панелі. Цей інтерфейс дозволяє користувачам контролювати умови мікроклімату, переглядати дані з сенсорів у режимі реального часу, а також керувати налаштуваннями системи.

Структура бази даних системи призначена для ефективного зберігання та управління зібраними даними з сенсорів та іншої релевантної інформації. Нижче наведено опис типової структури бази даних для комп'ютерної системи моніторингу мікроклімату виробничих приміщень:

- Колекція: Колекція - це група документів у MongoDB, подібно до таблиці в реляційній базі даних. Система може мати кілька колекцій для зберігання різних типів даних, пов'язаних з моніторингом мікроклімату.

- Документ: Документ - це запис у MongoDB, представлений як JSON-подібний об'єкт. Кожен документ у колекції відповідає певній сутності або точці даних у системі (рис. 2.2).

- Поля: Поля - це пари ключ-значення в документі, які містять фактичні дані. У контексті моніторингу мікроклімату поля можуть включати.

Дані з сенсорів. Для кожного параметру (температура, вологість, освітлення, co2) створена власна колекція, яка містить наступні поля: зону у якій розташовані сенсори, дата та час, унікальний індекс та значення параметру.

```
_id: ObjectId('64207648633a32e62bf14bf7')
area: "QA"
date: 2023-03-25T22:00:00.000+00:00
humidity_level: 92.3
```


Рис. 2.2. Приклад Документу

Параметри керування системою. Дана колекція містить налаштування для кожної зони, наприклад: мінімальні та максимальні значання параметрів, назву зони, булеве значення для окремих механізмів, а також унікальний номер (рис. 2.3).


```
_id: ObjectId('6423273a9eb823da1459eb37')
area_name: "AA"
co2_level_min: 440
co2_level_max: 600
temperature_min: 18.3
temperature_max: 25
humidity_min: 60
humidity_max: 80
temperature_auto: true
cool_is_on: false
heat_is_on: false
humidifier_is_on: true
fan: true
light_usage: 35
```


Рис. 2.3. Документ Налаштувань зони

Інформація про користувача. Це базові дані користувача, які використовуються під час авторизації до системи, такі як: пошта, пароль, ім’я, прізвище, посада (рис. 2.4)

```
_id: ObjectId('6453e17c8b7f9d738301c696')
email: "test@test.com"
password: "123456"
first_name: "Nina"
last_name: "Prokhorenko"
work: "Administrator of SDA"
```


Рис. 2.4. Дані користувача

Структура бази даних в MongoDB є гнучкою і не має схем, що дозволяє динамічно змінюватись і адаптуватись по мірі розвитку системи[17]. Це дає змогу ефективно зберігати та знаходити дані з сенсорів, інформацію про користувачів та параметри керування системою. Документно-орієнтована природа MongoDB добре узгоджується з гнучким характером даних моніторингу мікроклімату, що робить її підходящим вибором для управління та запитів до зібраної інформації.

2.5 Обґрунтування та організація вхідних та вихідних даних програми

Вхідними даними у створеній системі є ті, що вводяться користувачум через графічнйи інтерфейс, наприклад: поля вводу, перемикачі. Також, до таких даних належать дані отримані із сенсорів, тобто показники фізичного середовища (температура, вологість, освітлення, co2).

Вихідними данними вважаються: графіки - візуалізовані дані, які показують динаміку змін показників протягом певного часу; система сповіщення - листи іх звітами, що надходять користувачу у заданий період на електронну пошту.

2.6 Опис розробленої системи

2.6.1.Використані технічні засоби.

У розробці даної системи вистачає мінімальних ресурсів комп'ютера. Для роботи потрібні:

- персональний комп'ютер чи ноутбук,
- клавіатура;
- миша.

2.6.2 Використані програмні засоби

Для перевірки і тестування змодельованої мережі сенсорів та виконавчих механизмів використовувалося ПЗ Cisco Packet Tracer - це інструмент візуальної симуляції, який дозволяє створити топологію системи та протестувати її у змодельованому середовищі.

Для використання веб-застосунку стабільно працює на комп'ютерному обладнання з операційною системою Windows XP, 7, 8 чи 10 32 чи 64-розрядних версій. Також, має бути встановлено будь-який сучасний браузер.

2.6.3 Виклик та завантаження програми

Для запуску демонстраційної моделі необхідно виконати файл model.pkt, після чого можна протестувати та переглянути роботу даної системи.

Запуск веб-застосунку відбувається у браузері, після чого користувача буде перенаправлено до сторінки авторизації.

2.6.4 Опис моделі системи

Нижче буде описано демонстраційну модель, створену за допомогою ПЗ Cisco Packet Tracer, а також алгоритми збору даних для кожного з показників та роботи виконавчих пристроїв. На рис. 2.5 зображено роботу сенсорів та виконавчих механізмів для параметра температури.

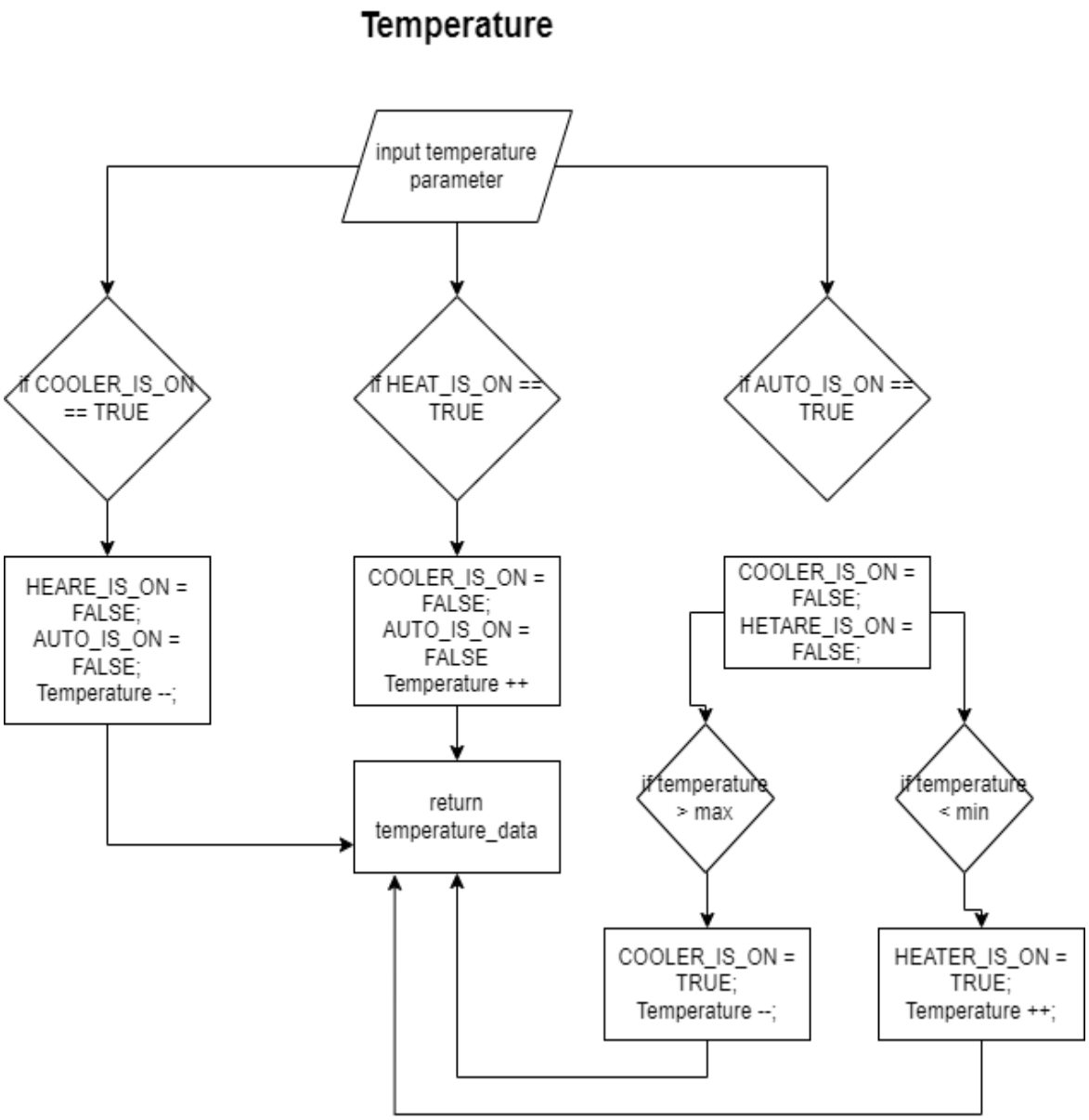

Рис. 2.5. Алгоритм роботи для показника “Температура”

Нижче наведено алгоритми для показників CO2, Вологості, Освітлення приміщення відповідно на рисунках 2.6 , 2.7 , 2.8.

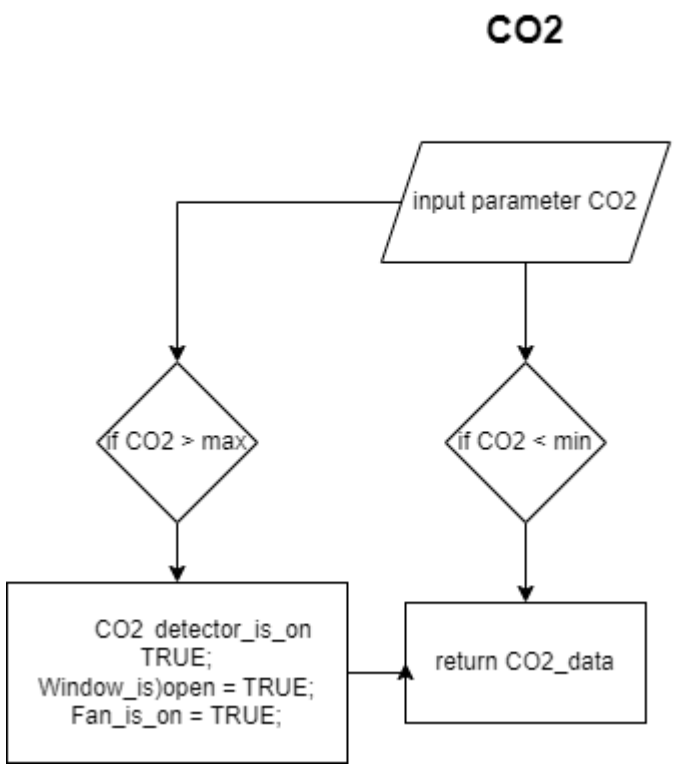

Рис. 2.6. Алгоритм роботи для показника “CO2”

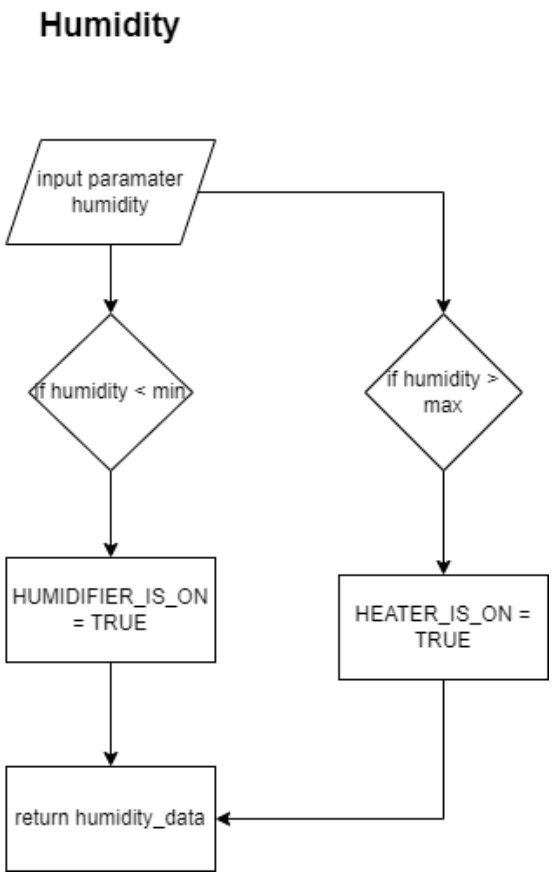

Рис. 2.7. Алгоритм роботи для показнка “Вологість”

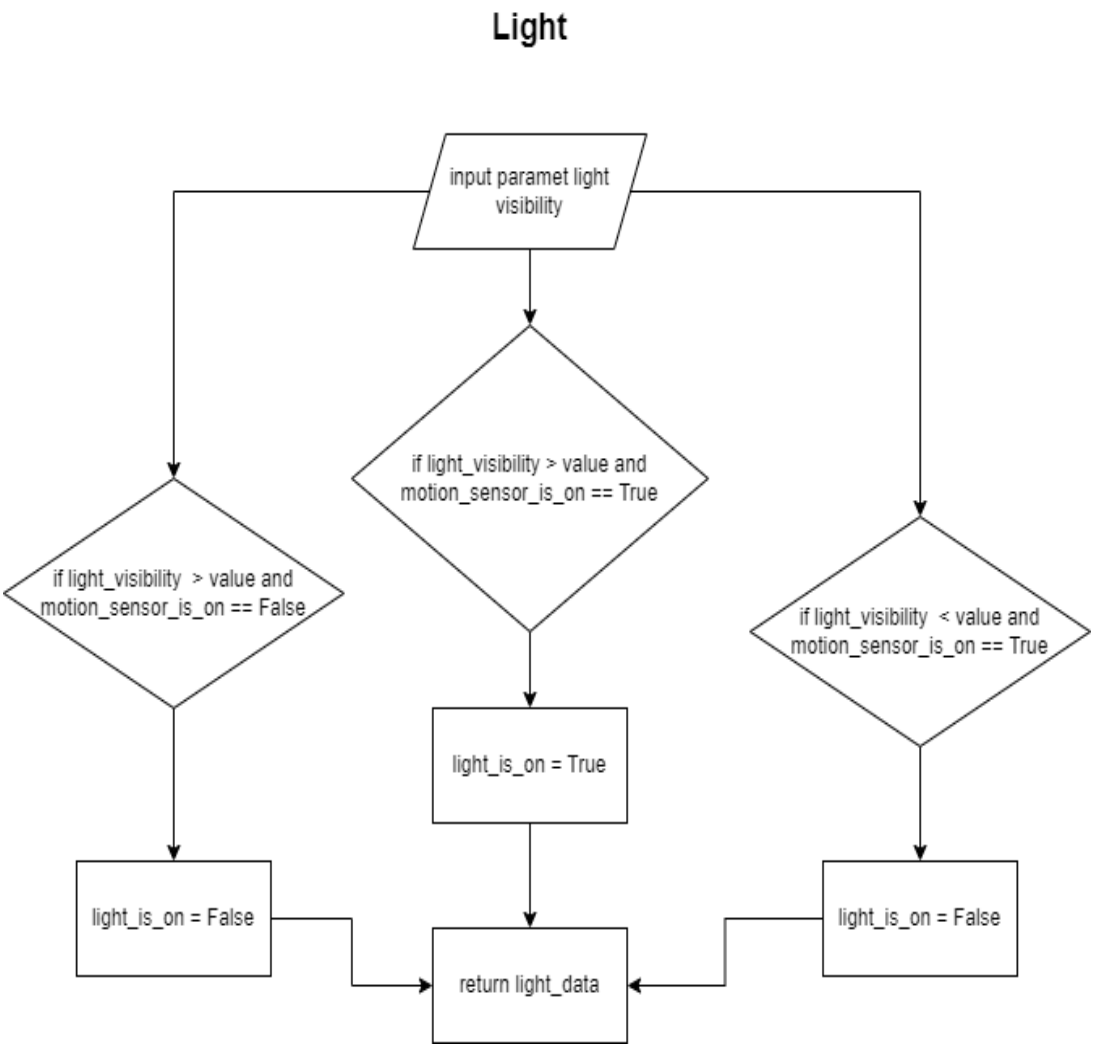

Рис. 2.8. Алгоритм роботи для показник “Освітлення”

Перед розробкою топології, окреслимо умовний план приміщення, у якому будуть розташовані компоненти системи та додамо позначення для деталізації (рис. 2.9)

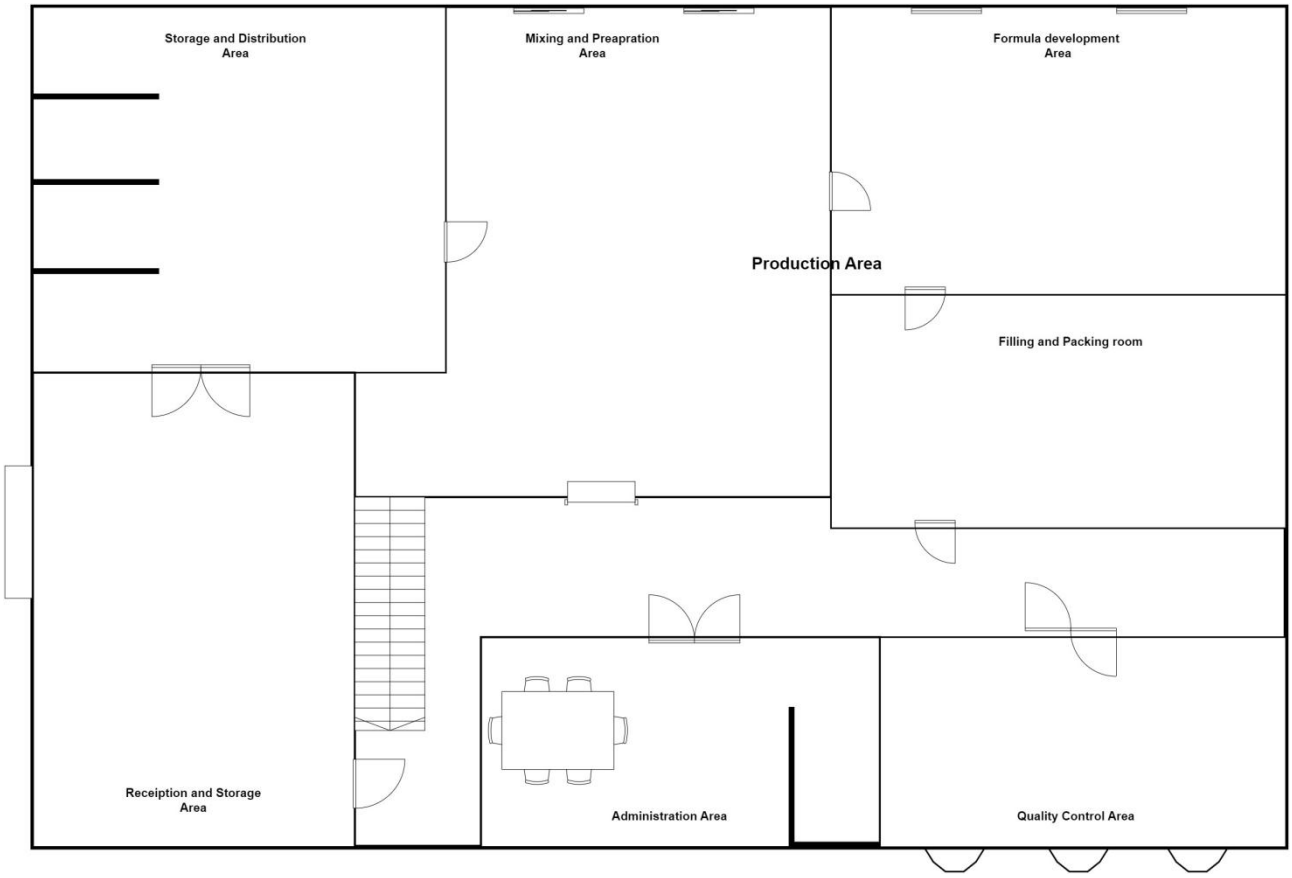

Рис. 2.9. План приміщення

Це виробниче приміщення, яке поділено на окремі зони, умови показників кожної із зон можна переглянути в таблиці 2.1[9,12,16]. Кожна з яких має власний набір сенсорів та виконавчих пристроїв, які регулюють умови мікроклімату. До них можуть застосовуватися різні налаштування. Наприклад, параметри температури та вологості у “Зоні Зберігання” не можуть співпадати із “Адміністративною Зоною”, так як для першої важливі умови зберігання - прохолода, сухість приміщення аби сировина довше була придатна до використання, а для другої такі параметри не завжди комфортні для персоналу.

Розмістивши потрібні компоненти їх потрібно з’єднати та утворити мережу (рис. 2.10).

Таблиця 2.1

Встановленні норми показників для приміщення

Зона	Температура, С	Вологість, %	Освітленність, %	Вуглекислий газ, ppm
Адміністративна	22-25	40-60	40	400
Виробнича	20-23	40-60	60	400
Перевірка якості	20-23	40-60	60	400
Прийому та зберігання	15-20	40-60	30	400-600
Розподілу	15-20	40-60	30	400-600

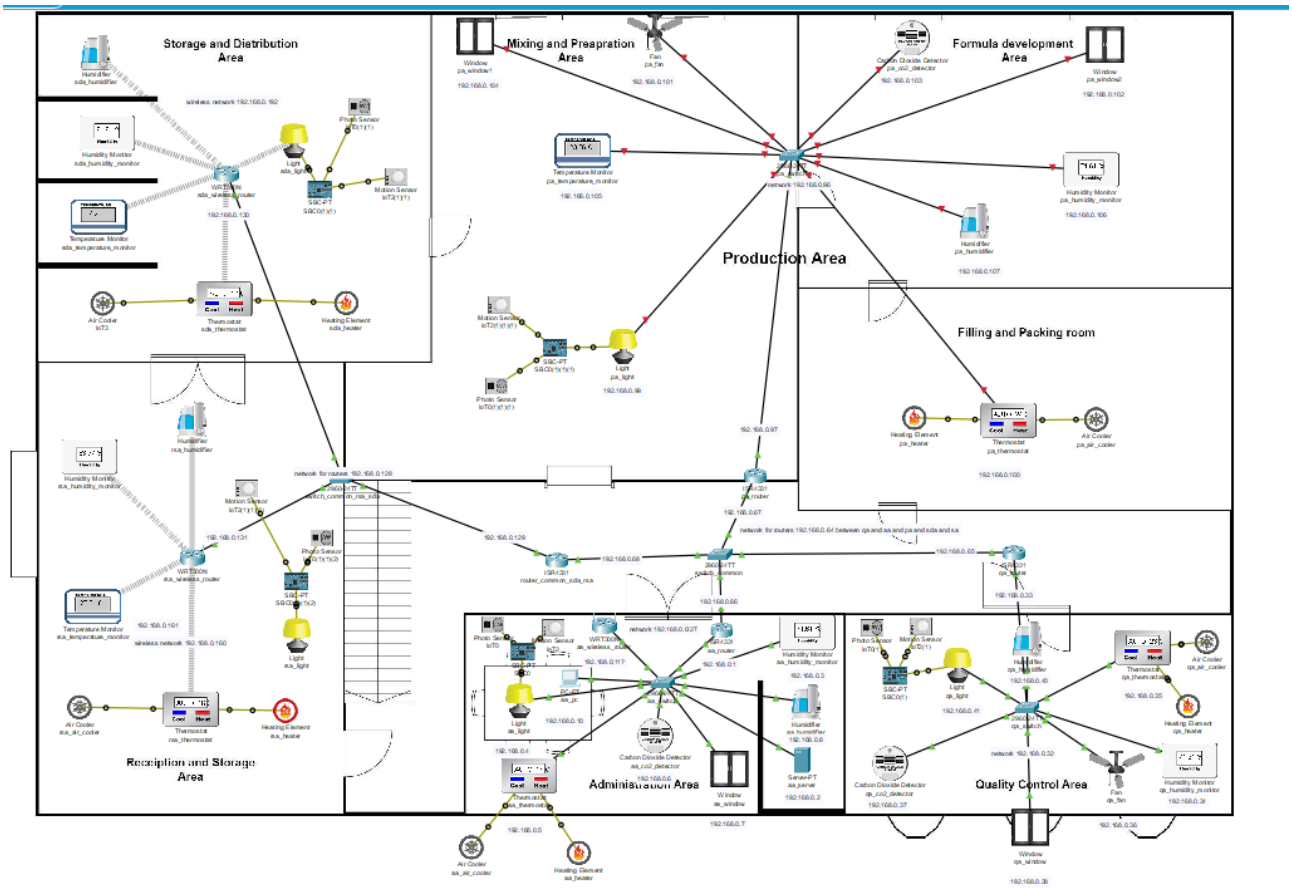

Рис. 2.10. Утворена мережа

У даній мережі представлено дротове та бездротове з’єднання пристроїв базуючись на вимогах приміщень [7,8].

Для встановлення зв'язку між різними компонентами та пристроями в системі, ієрархічна топологія мережі представляється відповідним рішенням. Застосовувана методологія передбачає використання декількох рівнів комутаторів і маршрутизаторів, що сприяє оптимальному розподілу трафіку і підвищує простоту управління.

На верхньому рівні може бути присутній центральний комутатор або маршрутизатор, який слугує для з'єднання всіх підпорядкованих комутаторів. Цей пристрій відповідає за передачу пакетів даних між різними сегментами мережі.

Кожна локальна мережа повинна мати окремі комутатори доступу, які пов'язані з основним комутатором/маршрутизатором. Комутатори доступу беруть на себе функції забезпечення підключення пристроїв та інших кінцевих пристроїв, таких як персональні комп'ютери (ПК) і сервери, в межах відповідних доменів.

Нижче наведено приклад можливого взаємозв'язку між різними областями та пристроями:

Виробнича зона може бути обладнана власним комутатором доступу, призначеним для підключення різних пристроїв та інших кінцевих пристроїв до цього сегменту мережі. Цей комутатор може бути підключений до основного комутатора/маршрутизатора, розташованого на найвищому рівні.

Зона якості може бути сконфігурована аналогічним чином за допомогою власного комутатора доступу і супутніх пристроїв, які також з'єднані з основним комутатором/маршрутизатором.

Адміністративна зона може бути обладнана окремим комутатором доступу, але вона також може бути з'єднана з основним комутатором/маршрутизатором за допомогою окремого маршрутизатора, призначеного для посилення заходів безпеки та сегментації мережевого середовища.

Забезпечення виділеного комутатора доступу для зони прийому та зберігання з підключенням до основного комутатора/маршрутизатора є потенційним рішенням, яке слід розглянути.

Зона зберігання і розподілу може бути обладнана декількома комутаторами доступу, один з яких спеціально призначений для пристроїв Інтернету речей, а решта комутаторів призначені для інших кінцевих пристроїв. Ці комутатори можуть встановлювати з'єднання з основним комутатором/маршрутизатором через проміжний маршрутизатор, забезпечуючи таким чином сегрегацію і безпеку пристроїв.

У створеній системі можна змоделювати параметри навколишнього середовища, і таким чином протестувати систему (рис. 2.11).

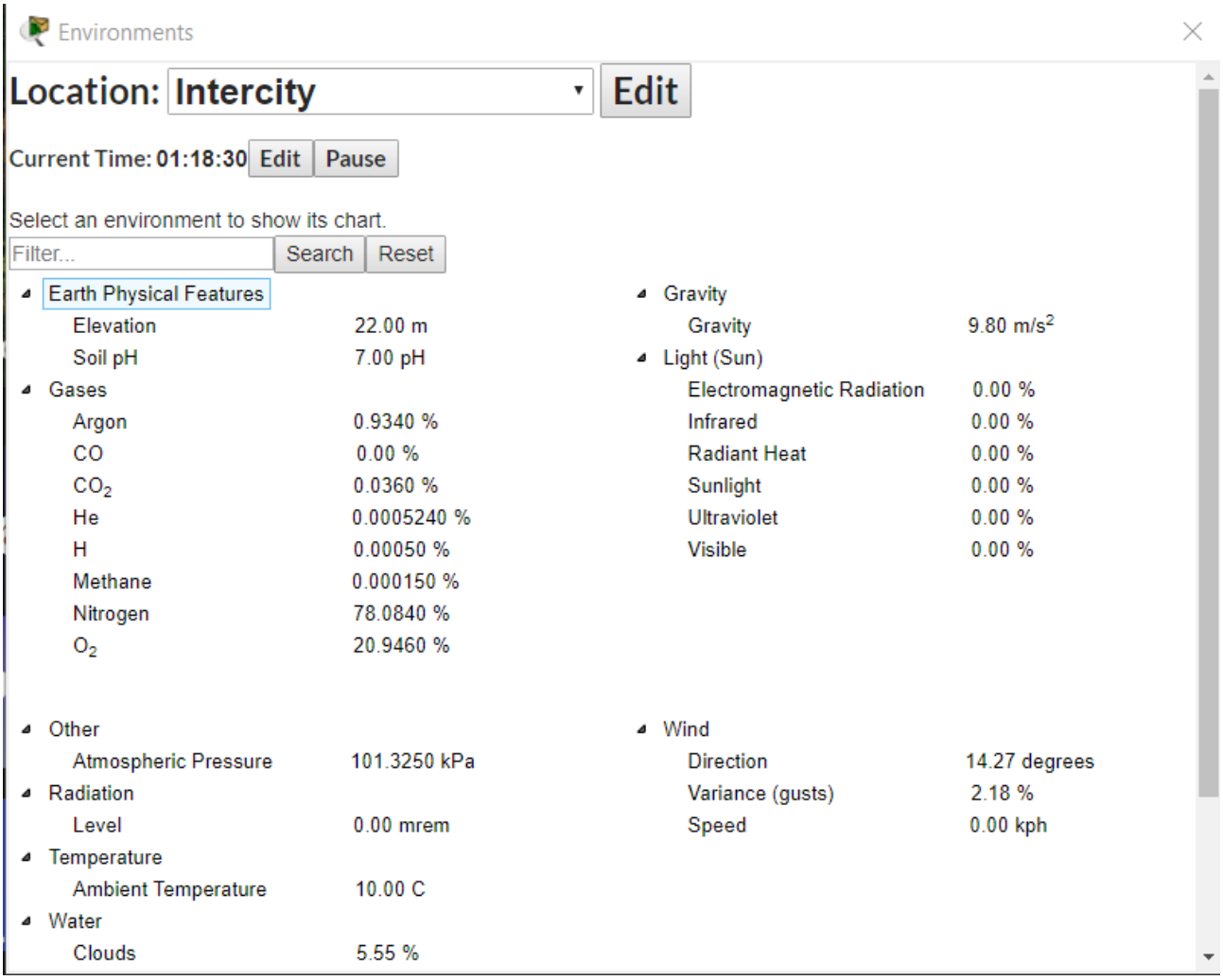

Рис. 2.11. Параметри середовища

У створеній моделі можна переглянути поточний стан кожного з параметрів завдяки сенсорам, які збирають інформацію. І відслідкувати зміни протягом певного часу для подальшого аналізу (рис. 2.12).

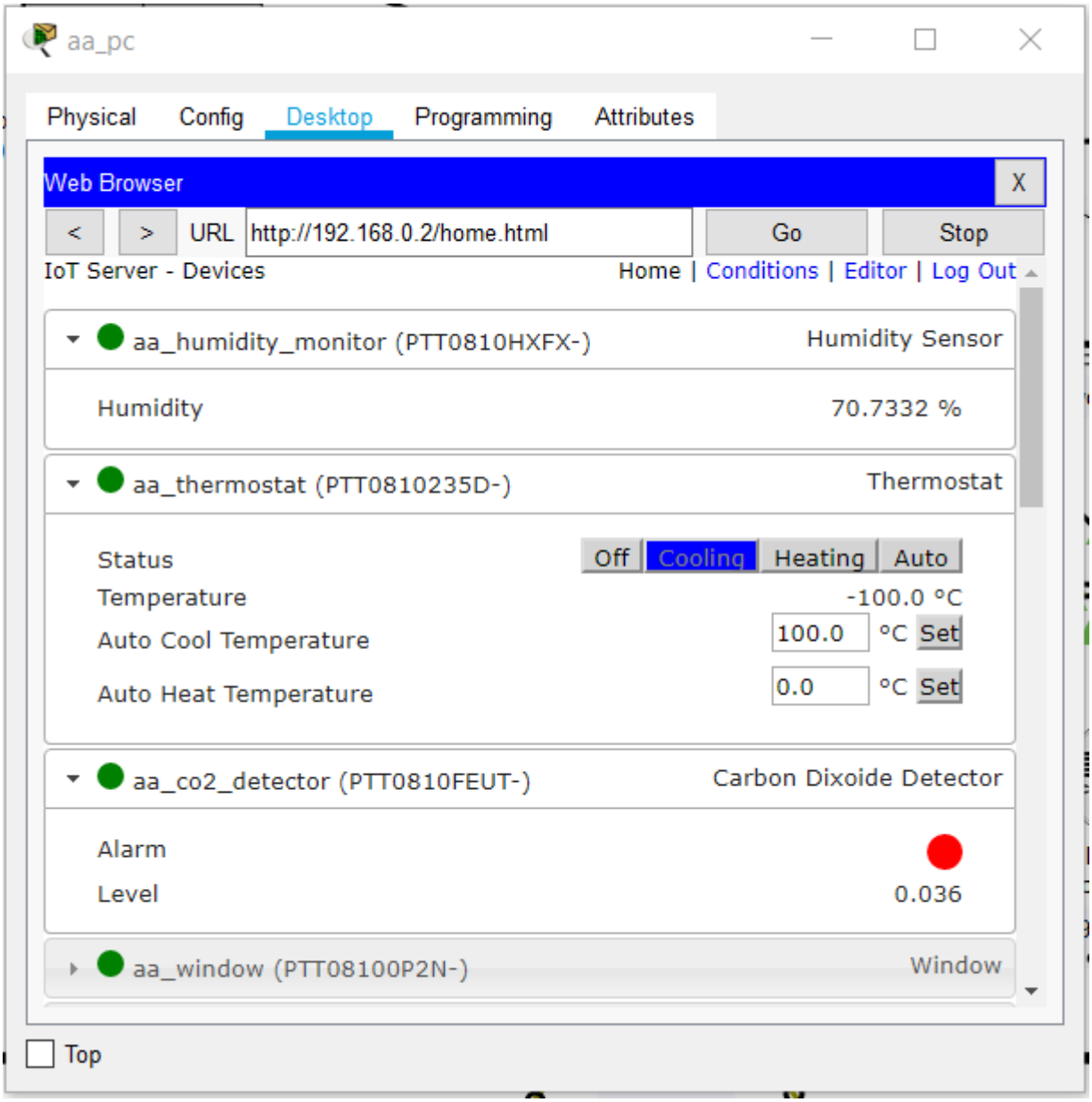

Рис. 2.12. Інтерфейс з параметрами

Також, є можливість налаштовувати виконавчі механізми (рис. 2.13) визначивши умови та встановити порогові значення (рис. 2.14) для кожного з показників[10,11,19]. Вихід за межі цього значення означатиме запуск одного або декількох виконавчих механізмів аби вирівняти ці показники у встановлені межі.

У прикладі наведеного нижче зображено встановлення умови увімкнення виконавчого механіма - зволожувача повітря. Механізм починає працювати на основі отриманих даних із сенсора: якщо отриманий показних менший за бажаний, то в такому випадку зволожувач змінює свій статус - вмикається.

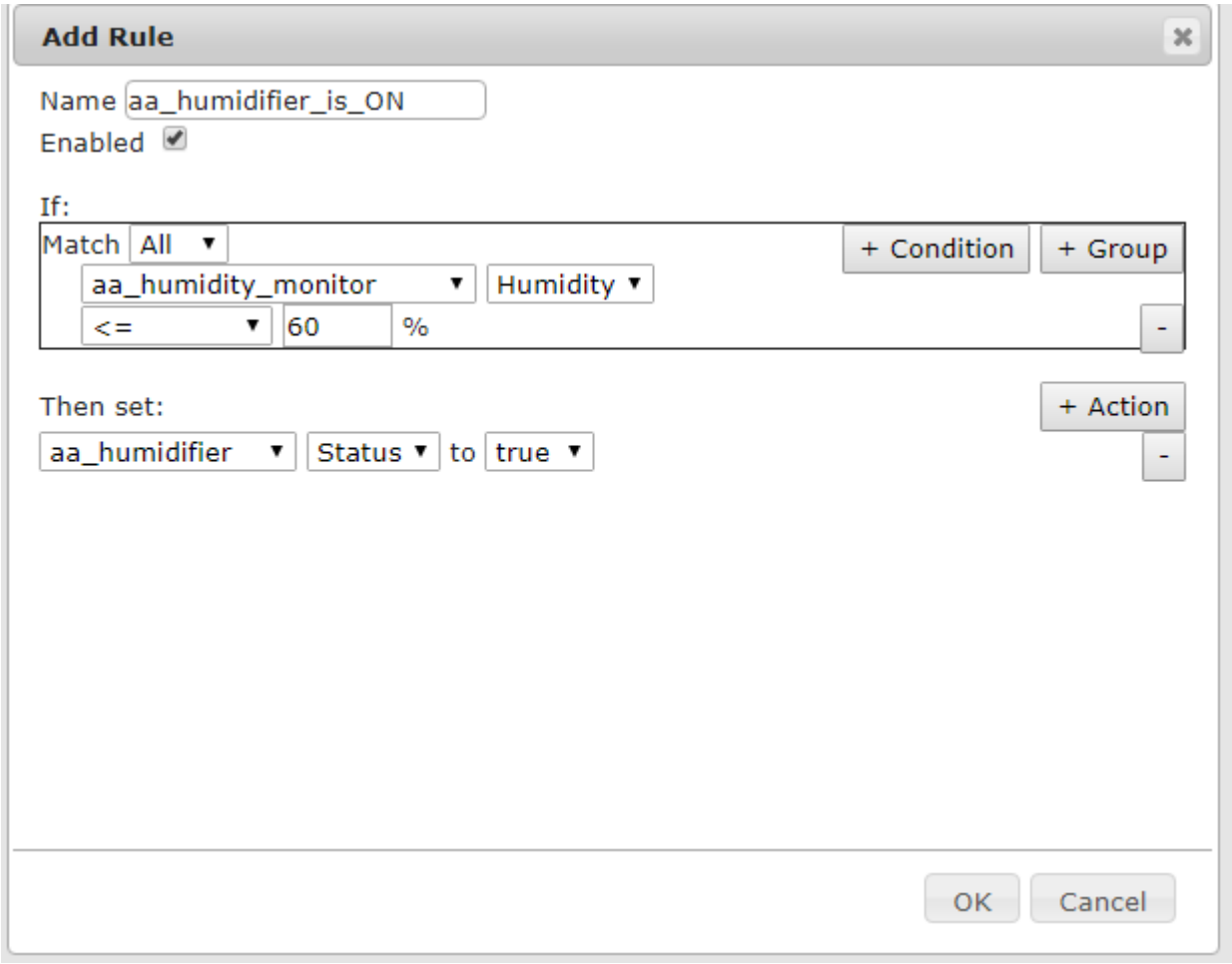

Рис. 2.13. Створення умови

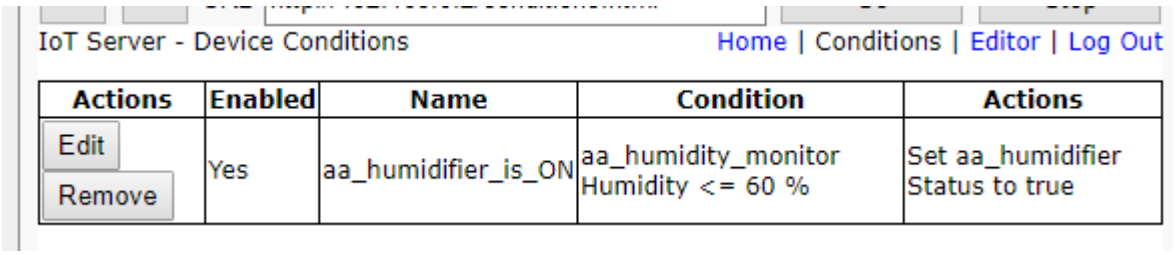

Рис. 2.14. Створена умова у списку

Перевірити роботу моделі можна задавши початкові показники фізичного середовища. На рисунку нище бачимо, задати початкові дані можна на графіку залежності часу від темеператури (рис. 2.15).

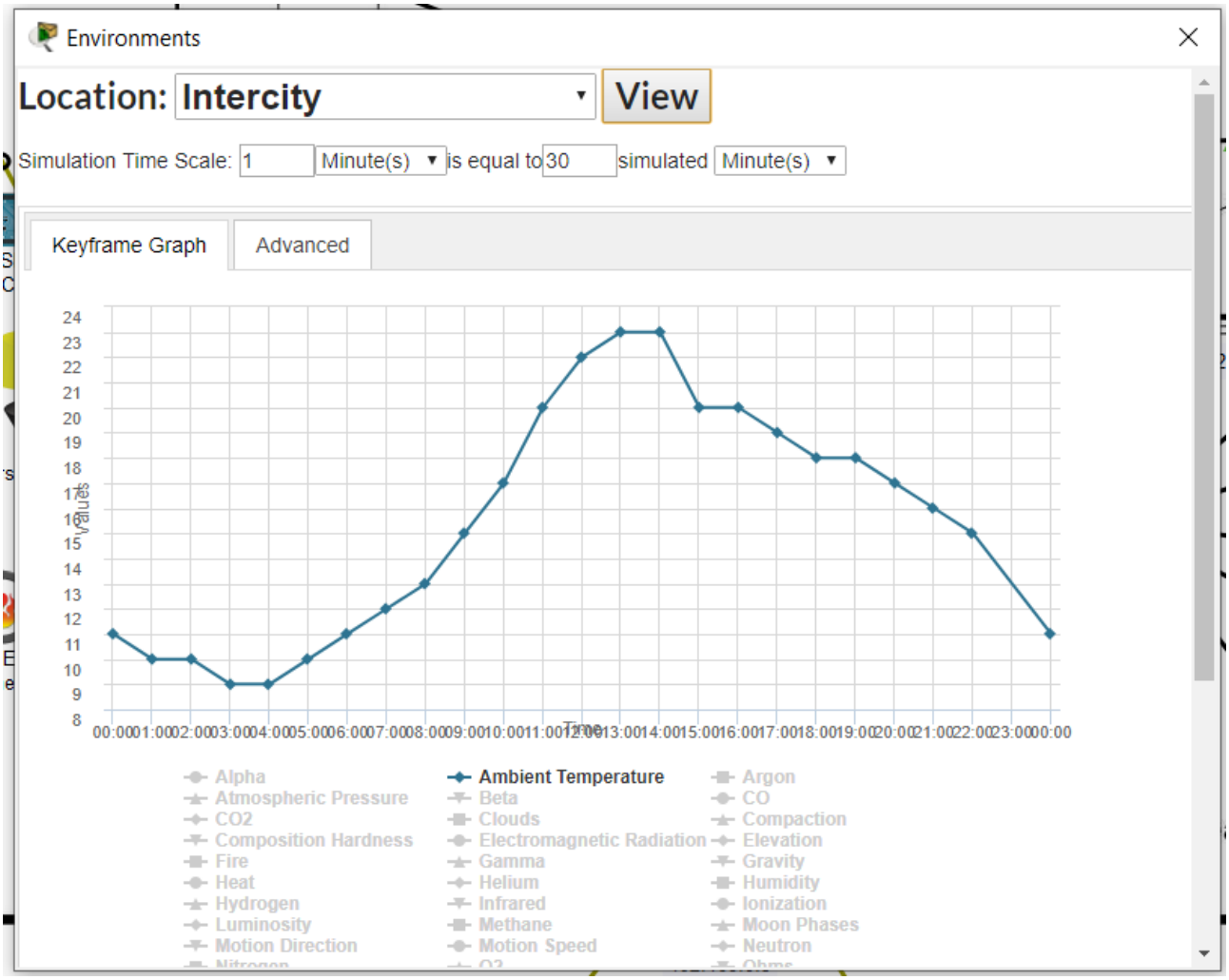

Рис. 2.15. Показник температури

Протягом дня температура зростає, але через налаштування із підвищенням температури сенсори отримають інформацію - перевищення значення показника температури у певній зоні, тому запускається виконавчий механізм у вигляді Air Cooler, яка працюватиме задля підтримки стабільної температури[14,15].

2.6.5 Опис інтерфейсу користувача

Так як ПЗ Cisco Packet Tracer не має можливості взаємодіяти із зовнішніми модулями (наприклад: бази даних), то для демонстрації графічного інтерфейсу користувача було написано код на мові програмування Python, який генеруватиме дані показників та записуватиме їх до БД.

Після запуску веб-додатку запускається стартова сторінка у вигляді форми логіну. Якщо сесія користувача не вичерпалася, тоді його автоматично перекидає на головну сторінку. Надалі, кожного разу, якщо час сесії користувача вичерпавався за встановлений період неактивності, то з поточної сторінки на якій перебував користувач його автоматично пересилатиме на сторінку логіну (рис. 2.16).

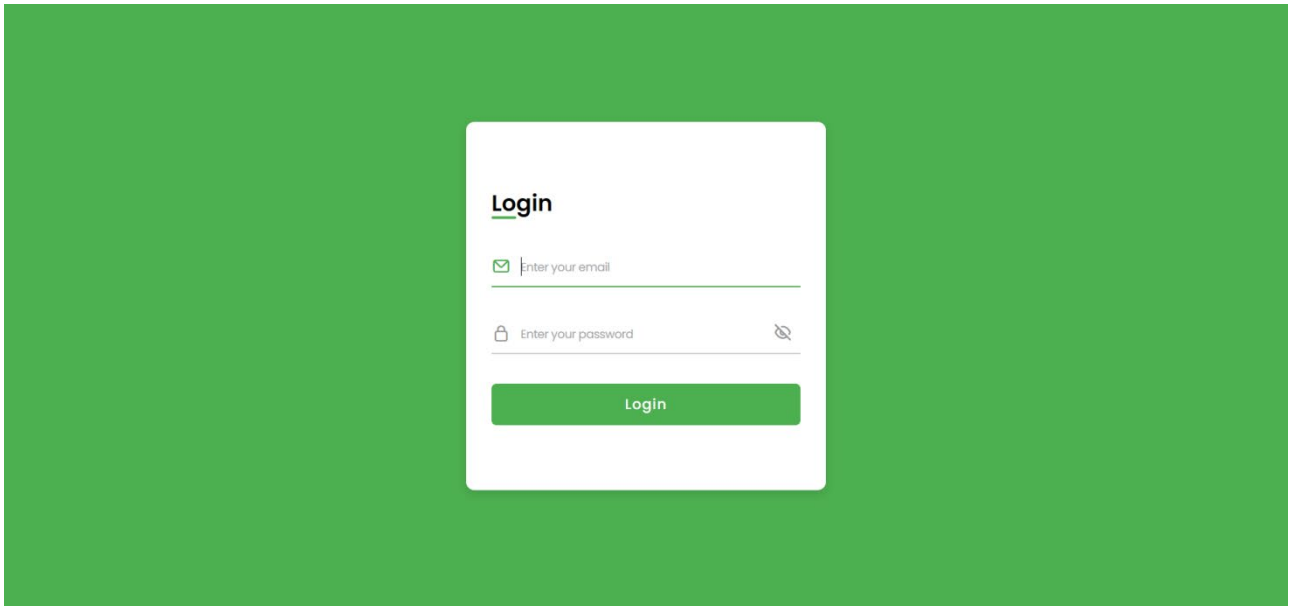

Рис. 2.16. Форма входу користувача

У разі неправильно введеного корстувачем паролю або пошти з’являється повідомлення про помилку з контекстом цієї помилки. Якщо це неправильний пароль, то повідомлення про помилку містить нступний текст: “Wrong password provided”, а для неіснуючої пошти “No such user in our system. Please check your credentials” (рис. 2.17).

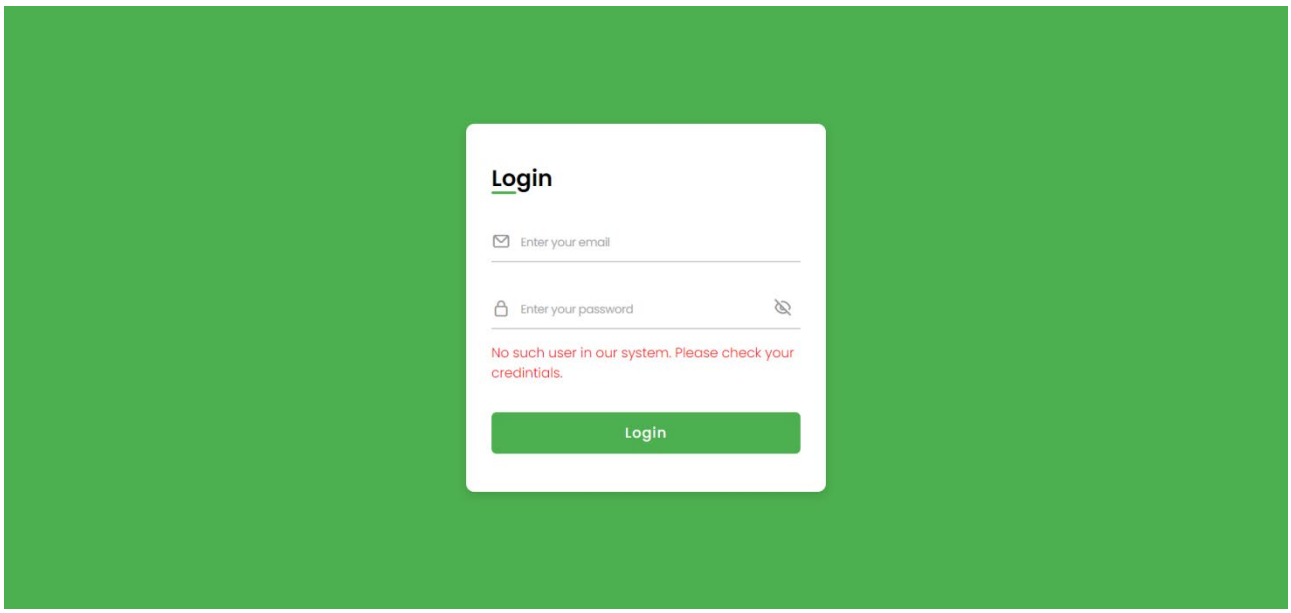

Рис. 2.17. Повідомлення про помилку входу в систему

Після успішного входу в систему користувач переходить до головної сторінки і за допомогою навігаційного меню переходить до потрібної сторінки веб-додатку:

- сторінка профілю користувача;
- звіт із поточними даними кожної із зон;
- головна сторінка;

У профілі користувача (рис. 2.18) міститься його головна інформація: посада, ім’я та пошта. За бажанням дані можна оновити. (рис.2.19)

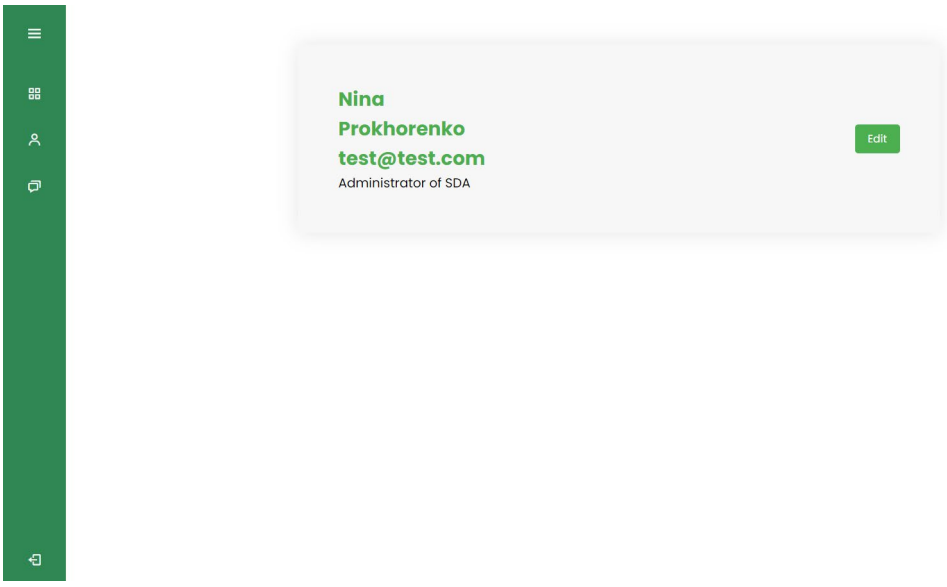

Рис. 2.18. Профіль користувача

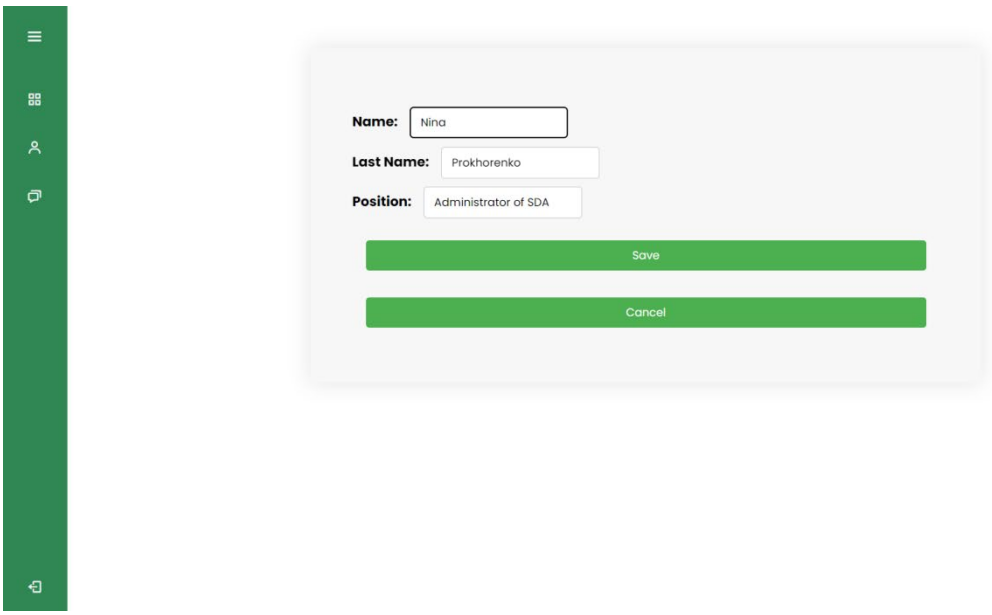

Рис. 2.19. Оновлення даних користувача

На сторінці звіту із поточними даними користувач переглядає інформацію щодо параметрів та виконавчих механізмів (рис. 2.20). Також, є можливість відслідковування даних на графіку (рис. 2.21).

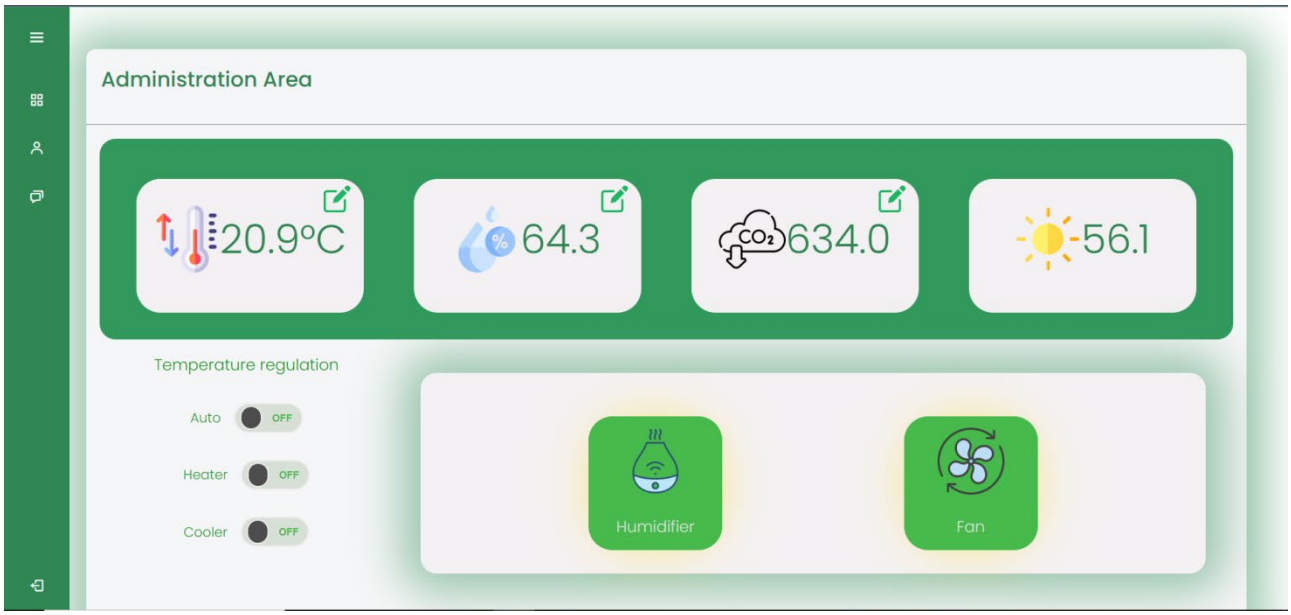

Рис. 2.20. Сторінка звіту

Рис. 2.21. Графік

Для налаштування граничних значень є форма вводу даних для кожного параметру (рис. 2.22, рис. 2.23, рис. 2.24), якщо введені дані містять неправильний формат користува отримає повідомлення про помилку (рис. 2.25).

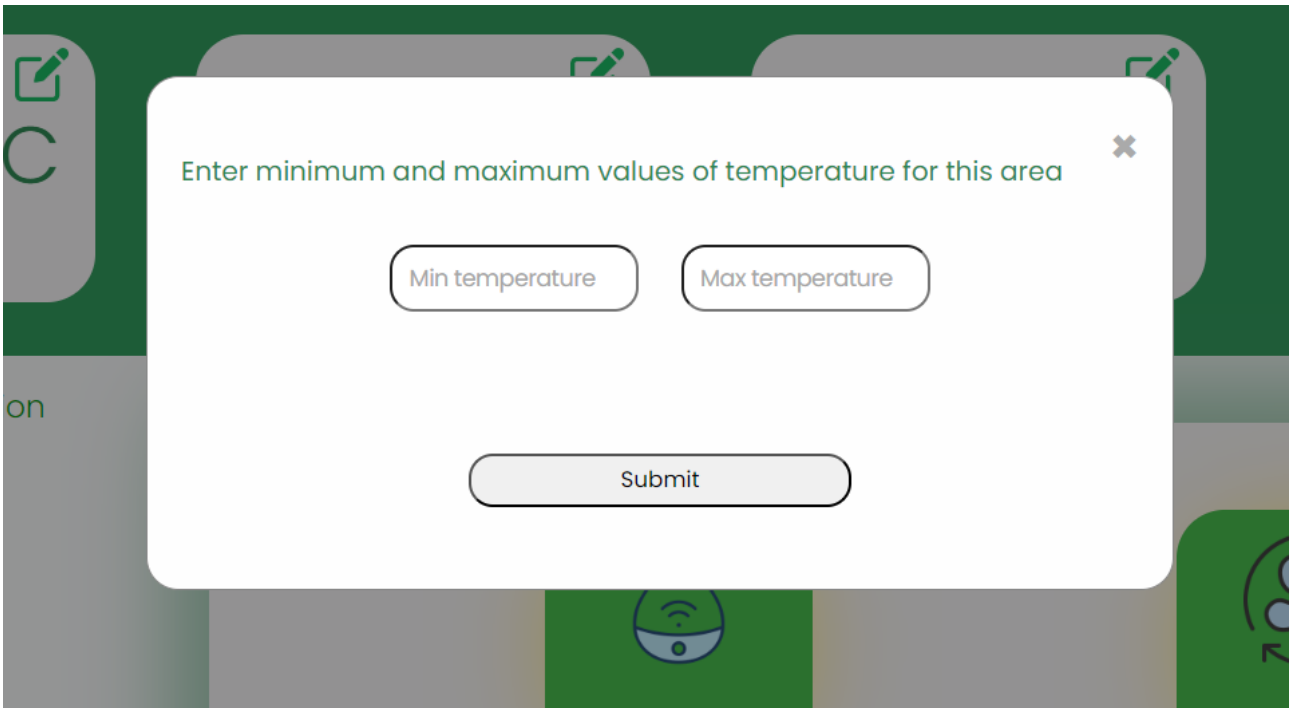

Рис. 2.22. Налаштування Температури

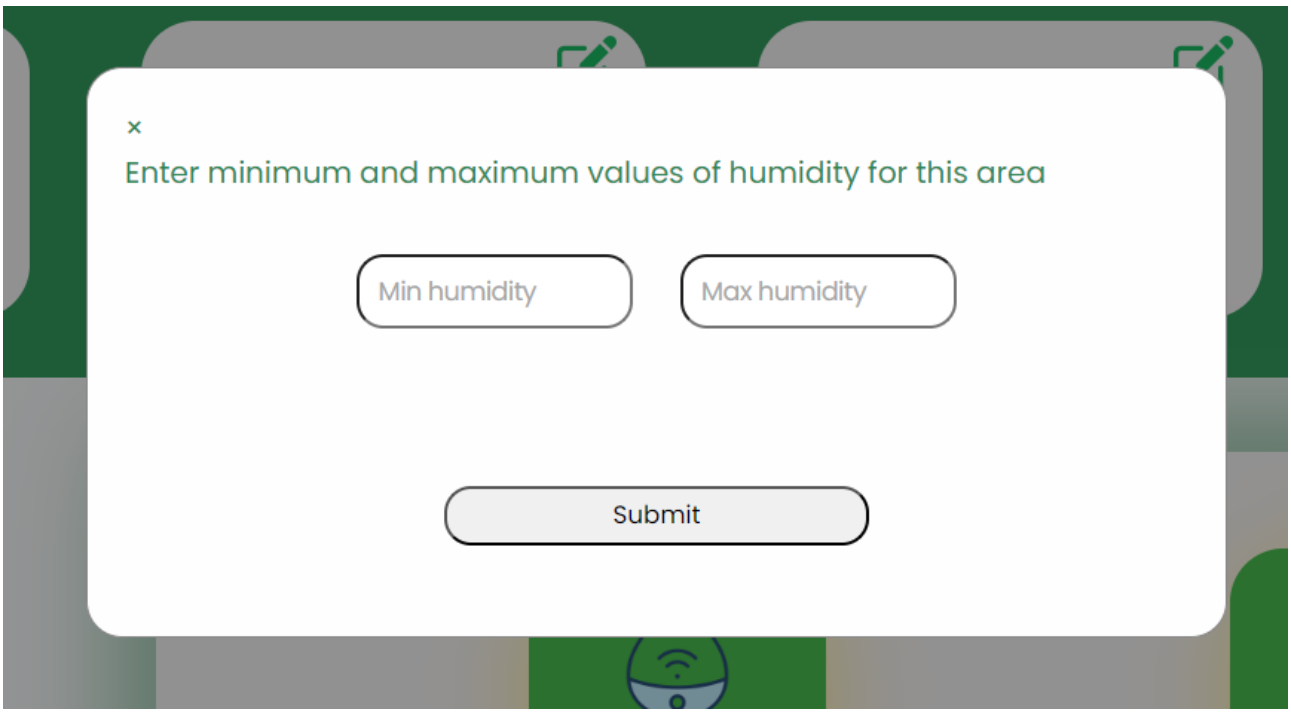

Рис. 2.23. Налаштування Вологості

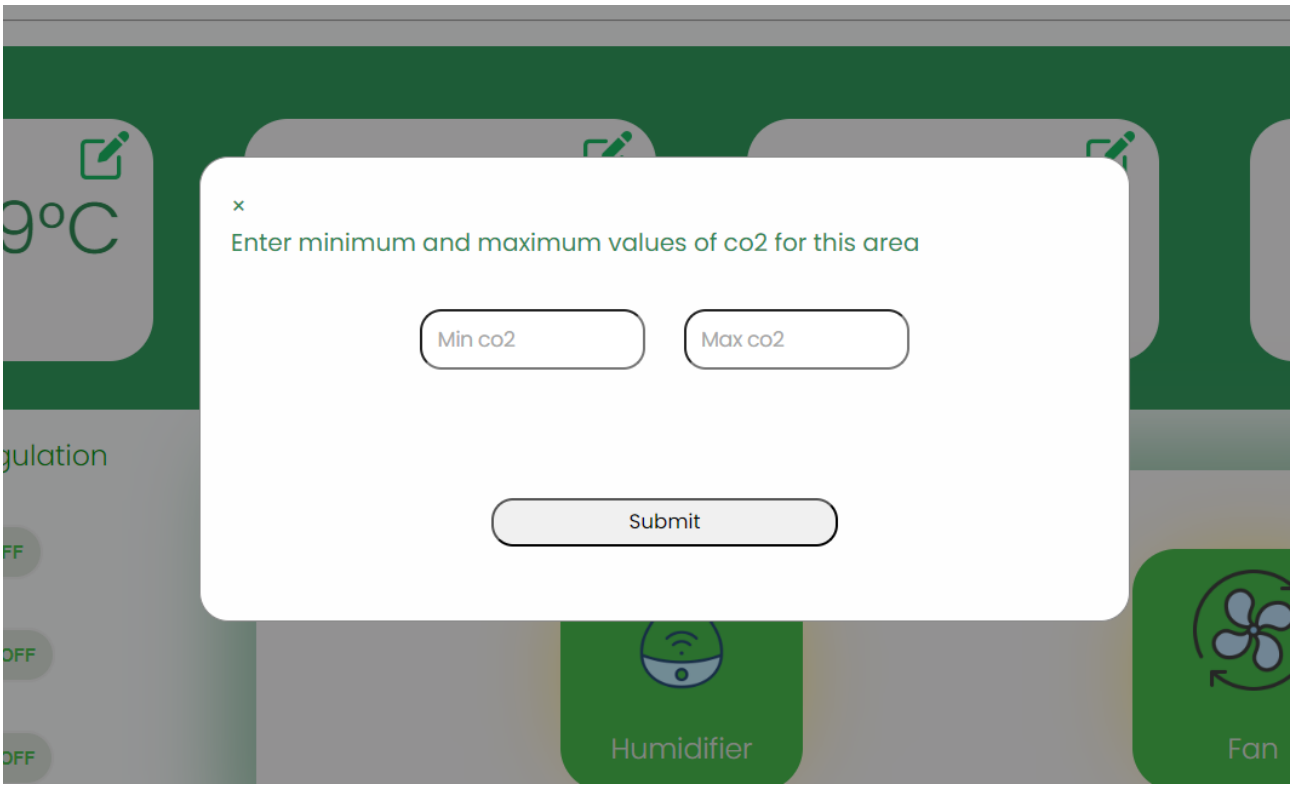

Рис. 2.24. Налаштування CO2

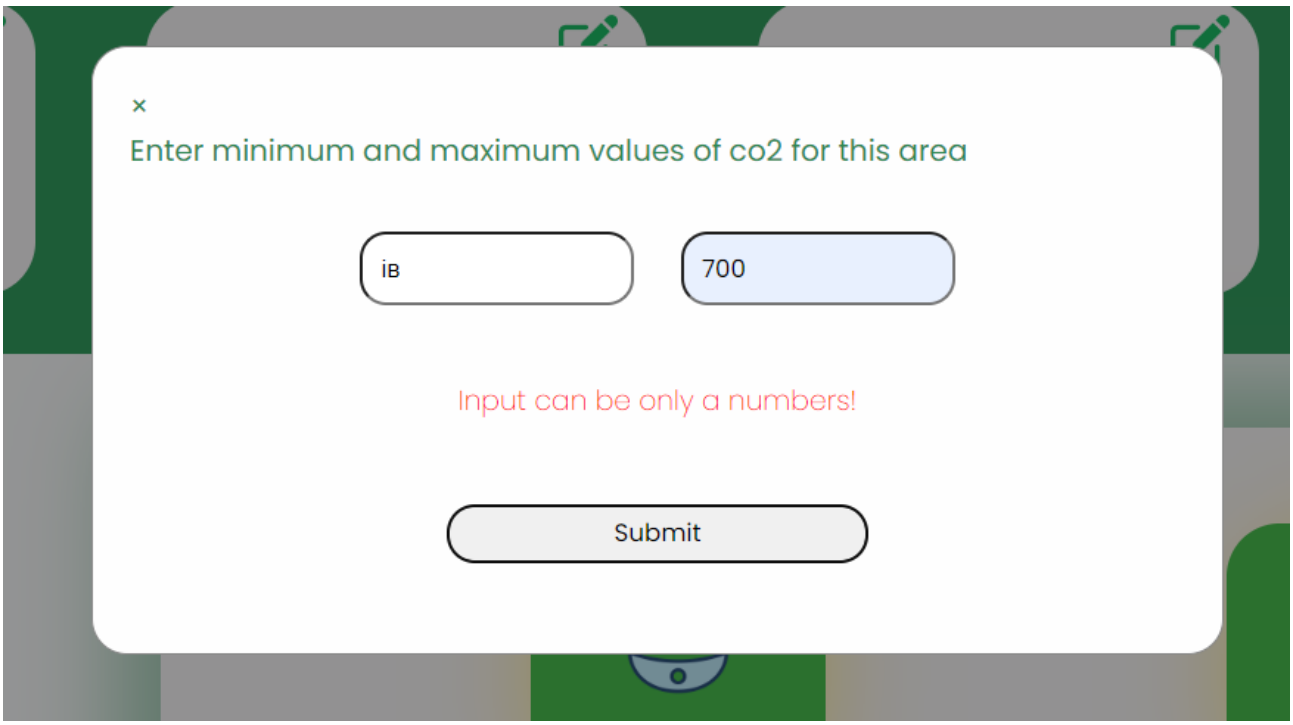

Рис. 2.25. Повідомлення про помилку

Сповіщення про поточний стан параметрів надсилаються користувачеві кожні три години у наступному вигляді. Таке сповіщення на вказану електронну пошту містить дату, час та показники із кожного приміщення (рис. 2.26).

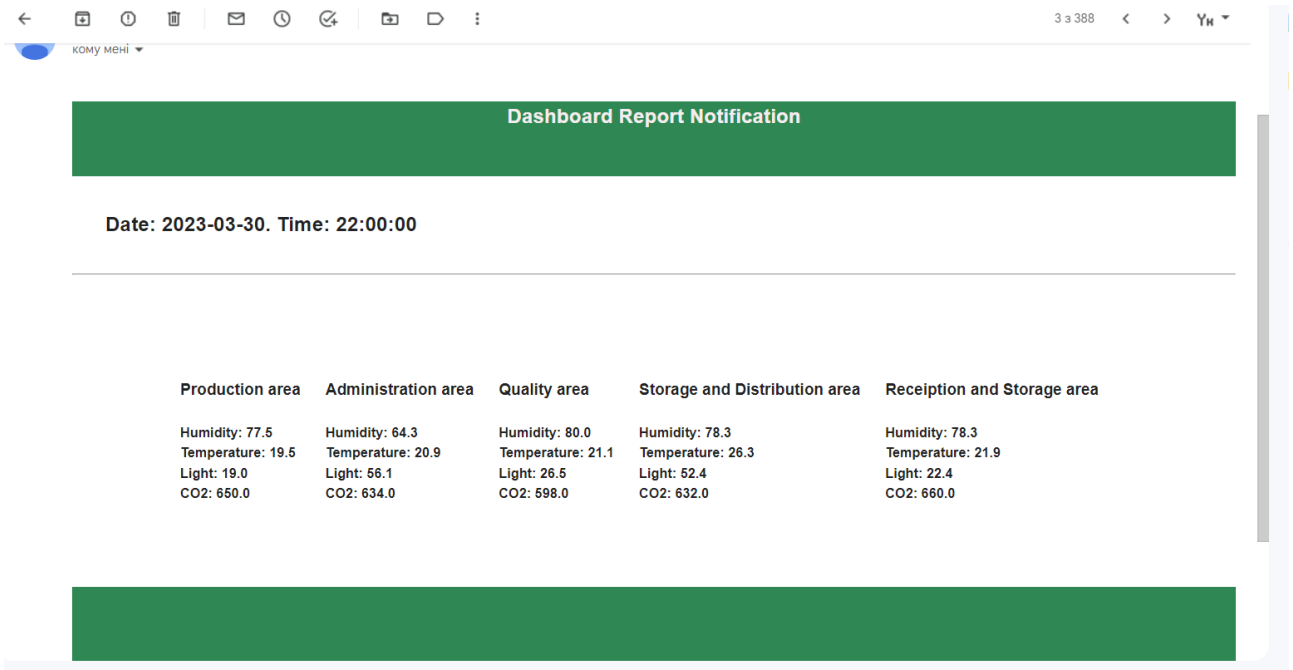

Рис. 2.26. Сповіщення

РОЗДІЛ 3

ЕКОНОМІЧНИЙ РОЗДІЛ

Вихідні дані для розробки програмного забезпечення мають такі характеристики:

1) Передбачуване число операторів: 3100.

2) Коефіцієнт складності програми: 1,4.

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

4) Середня годинна заробітна плата програміста: 104 грн/год [13]

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

Вартість машино-години для ЕОМ може включати різні компоненти. Однак, основними факторами, які можуть впливати на ціну, є наступні:

- Амортизація: Вартість ЕОМ може включати амортизацію обладнання.

- Електроенергія: Робота ЕОМ вимагає електричної енергії. Вартість електроенергії може бути розрахована на основі витрат на кіловат-годину (кВт-год). ЕОМ споживає 2 кВт-год на годину роботи, а вартість одного кВт-год становить 1,63 грн, то витрати на електроенергію для однієї машино-години складуть 1,63 кВт-год * 2 грн/кВт-год = 3,26 грн.

- Обслуговування та технічна підтримка: Цей елемент може включати витрати на обслуговування та ремонт ЕОМ, а також оплату фахівців, які забезпечують технічну підтримку. Вартість може бути розрахована на основі погодинної ставки технічного персоналу та середнього часу, який вони витрачають на обслуговування кожної машини.

3.1 Розрахунок трудомісткості та вартості розробки програмного продукту

Ускладнення процесу нормування праці під час створення програмного забезпечення пояснюється творчим характером роботи програміста. Внаслідок

цього, для оцінки трудомісткості розробки ПЗ може бути використана система моделей з різним рівнем точності.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,} \qquad (3.1)$$

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

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

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

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

$t_д$ – витрати праці на підготовку документації.

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

Умовне число операторів (підпрограм):

$$Q = q \cdot C(1 + p), \text{ де} \qquad (3.2)$$

q – передбачуване число операторів;

C – коефіцієнт складності програми;

p – коефіцієнт кореляції програми в ході її розробки.

$$Q = 3100 \times 1{,}4(1 + 0{,}3) = 5642;$$

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

$$t_u = \frac{Q \cdot B}{(75...85)K}, \text{ людино-годин,} \tag{3.3}$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі, В=1,35;

K – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності становить 1,3;

$$t_u = \frac{5642 \cdot 1{,}35}{85 \cdot 1{,}3} = 68{,}93 \text{, людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25)K} \tag{3.4}$$

$$t_a = \frac{5642}{25 \cdot 1{,}3} = 173{,}6 \text{, людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K} \tag{3.5}$$

$$t_n = \frac{5642}{25 \cdot 1{,}3} = 173{,}6 \text{, людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)K} \tag{3.6}$$

$$t_{отл} = \frac{5642}{5 \cdot 1{,}3} = 868 \text{, людино-годин,}$$

- за умови комплексного налагодження завдання:

$$t^{к}_{отл} = 1{,}2 \cdot t_{отл} \tag{3.7}$$

$$t^{к}_{отл} = 1{,}2 \cdot 868 = 1041{,}6 \text{, людино-годин}$$

Витрати праці на підготовку документації:

$$t_{Д} = t_{Др} + t_{До} \tag{3.8}$$

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

$$t_{Др} = \frac{Q}{(15...20)K} \tag{3.9}$$

$$t_{др} = \frac{5642}{20 \cdot 1{,}3} = 217 \text{, людино-годин.}$$

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

$$t_{до} = 0{,}75 \cdot t_{др} \qquad (3.10)$$

$t_{до} = 0{,}75 \cdot 217 = 162{,}75$, людино-годин.

$t_{д} = 217 + 162{,}75 = 379{,}75$, людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:

t = 50+68,93+173,6+173,6+868+379,75=1713,88 , людино-годин.

У результаті ми розрахували, що в загальній складності необхідно 1713,88 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $K_{по}$ включають витрати на заробітну плату виконавця програми $З_{зп}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн}, \tag{3.11}$$

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

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн}, \tag{3.12}$$

де t – загальна трудомісткість, людино-годин;

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

$$З_{ЗП} = 1713{,}88 * 104 = 178243{,}52, \text{грн.}$$

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

$$З_{МВ} = t_{отл} \cdot C_{М}, \text{ грн}, \tag{3.13}$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$C_{МЧ}$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 868 \cdot 10 = 8680, \text{ грн.}$$

$$K_{ПО} = 178243{,}52 + 8680 = 186923{,}52, \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ мес.} \tag{3.14}$$

де B_k - число виконавців;

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

$$T = \frac{1713{,}88}{1 \cdot 176} = 9{,}73 \text{ міс.}$$

Висновки. Час розробки даного програмного забезпечення складає 1713,88 людино-годин. Таким чином, очікувана тривалість розробки складе 9,73 місяця при 40 годинному робочому тижні (місячний фонд робочого часу 176 годин), а витрати на створення програмного забезпечення складатимуть 192635,04 грн.

ВИСНОВКИ

Розробка комп'ютерної системи моніторингу стану мікроклімату в промислових приміщеннях дала значні результати і продемонструвала високий ступінь новизни та інноваційності. Завдяки ретельному дослідженню, аналізу та впровадженню, система успішно вирішила проблеми, пов'язані з моніторингом та контролем мікроклімату в промислових умовах.

Практична цінність системи очевидна в її здатності забезпечувати моніторинг і контроль температури, вологості, якості повітря та інших параметрів навколишнього середовища в режимі реального часу. Забезпечуючи оптимальні умови праці, система сприяє підвищенню продуктивності, поліпшенню самопочуття працівників та підтримці безпечного і комфортного робочого середовища.

Порівняння з існуючими аналогами підкреслює явні переваги розробленої системи. На відміну від традиційних підходів до моніторингу, які покладаються на ручне втручання та періодичні перевірки, комп'ютерна система пропонує безперервний, автоматизований моніторинг та можливість негайного реагування. Це не тільки мінімізує ризик потенційних небезпек, але й дозволяє вчасно втручатися і вносити корективи для підтримання оптимальних умов мікроклімату.

З наукової точки зору, дослідження, проведені в рамках даної кваліфікаційної бакалаврської роботи, сприяли поглибленню знань в області моніторингу промислового мікроклімату. Проектування, впровадження та оцінка системи продемонстрували ефективність використання технологій Інтернету речей, аналізу даних та автоматизації в контексті управління мікрокліматом. Результати дослідження та використані методології можуть слугувати основою для майбутніх досліджень і розробок у цій галузі.

Економічний аналіз, проведений в рамках цієї роботи, надав цінну інформацію про трудомісткість та вартість, пов'язані з розробкою програмного продукту. Розрахунки та оцінки допомогли оцінити фінансову життєздатність,

розподіл ресурсів та потенційну рентабельність інвестицій. Оцінені трудомісткість і вартість надають зацікавленим сторонам важливу інформацію для прийняття рішень, бюджетування та управління проектом.

З точки зору прогнозних припущень, очікується, що розроблена комп'ютерна система буде продовжувати розвиватися і адаптуватися до нових технологій і галузевих вимог. Масштабованість і гнучкість системи дозволяють здійснювати майбутні вдосконалення, такі як включення передових алгоритмів машинного навчання для прогнозного аналізу та інтеграція з іншими системами промислової автоматизації. Система має потенціал для ширшого застосування в різних галузях промисловості, сприяючи поліпшенню робочого середовища та підвищенню операційної ефективності.

Таким чином, розробка комп'ютерної системи моніторингу мікроклімату в промислових приміщеннях досягла значних результатів і продемонструвала її практичну та наукову значимість. Здатність системи забезпечувати моніторинг в режимі реального часу, комп’ютеризоване управління та розширену аналітику відрізняє її від традиційних підходів. Економічний аналіз надав цінну інформацію про трудомісткість та вартість, що сприяло прийняттю обґрунтованих рішень. Завдяки своїм новим функціям, масштабованості та потенціалу для подальшого розвитку, система є цінним внеском у сферу моніторингу та управління промисловим мікрокліматом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Наслідки високих температур на організм людини: / URL https://www.who.int/news-room/fact-sheets/detail/climate-change-heat-and-health (дата звернення 12.03.2023)

2. Оптимальна вологість повітря: / URL https://www.robertbpayne.com/perfect-indoor-humidity-for-all-seasons/ (дата звернення 12.03.2023)

3. Вплив вуглекислого газу на людину: / URL https://www.epfl.ch/labs/hobel/impact-of-indoor-climate-on-human-health/ (дата звернення 12.03.2023)

4. Токсини CO2: / URL https://www.health.state.mn.us/communities/environment/air/toxins/co2.html (дата звернення 12.03.2023)

5. Industrial IoT: Improving Safety and Efficiency: / URL https://bridgera.com/industrial-iot-improving-safety-and-efficiency/ (дата звернення 01.04.2023)

6. Internet of Things Simon Cirani та ін.// John Wiley & Sons Ltd 2019, 403c (дата звернення 03.03.2023)

7. Sreenivas Eeshwaroju, Praveena Jakkula, Subramanian Ganesan An IoT based application to address safety concerns of an Individual, Group and an Entity // 2020 International Conference on Computing and Information Technology, University o f Tabuk, Kingdom of Saudi Arabia. (дата звернення 20.04.2023)

8. Ghaliya Alfarsi , Ragad M Tawafak , Abir Alsidiri, Jasiya Jabbar , Sohail Iqbal Malik, Maryam Alsinani // Using Cisco Packet Tracer to simulate Smart Home, Volume 08, Issue 12 (December 2019) IJERT, -ISSN (Online) : 2278-0181 (дата звернення 20.04.2023)

9. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98 N 7 від

10.12.98 : URL https://zakon.rada.gov.ua/rada/show/v0007282-98#Text (дата звернення 15.05.2023)

10. Oksana L. Korenivska , Tetiana M. Nikitchuk , Tetiana A. Vakaliuk , Vasyl B. Benedytskyi and Oleksandr V. Andreiev // IoT monitoring system for microclimate parameters in educational institutions using edge devices // Proceedings of the 3rd Edge Computing Workshop Zhytomyr, Ukraine, April 7, 2023. (дата звернення 15.05.2023)

11. Temperature Monitoring at Food Processing Plants with Cisco's Industrial Sensor Solution: URL https://www.cisco.com/c/en/us/products/collateral/cloud-systems-management/industrial-asset-vision/temp-monitor-industrial-sensor-so.html (дата звернення 16.04.2023)

12. Встановленні норми вуглекслого газу: URL https://aerostar.ua/ua/news/novosti/co2-u-primischennjahl-na-scho-vplivae-dioksid-vuglecju-u-povitri-ta-jak-zmenshiti-jogo-riven.html.. (дата звернення 12.05.2023)

13. Заробітна плата в IT сфері: URL https://www.work.ua/salary-it/ (дата звернення 05.06.2023)

14. Packet Tracer 7.x - Internet of Things tutorials: URL https://www.packettracernetwork.com/internet-of-things/ (дата звернення 20.04.2023)

15. IoT advanced programming & automation: URL https://www.packettracernetwork.com/internet-of-things/iot-advanced-programming.html (дата звернення 20.04.2023)

16. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень, N 42 від 01.12.99 м.Київ (дата звернення 21.05.2023)

17. MongoDB Documentation: URL https://www.mongodb.com/docs/ (дата звернення 12.05.2023)

18. Flask Documentation: URL https://flask.palletsprojects.com/en/2.3.x/ (дата звернення 12.05.2023)

19. Norman Gwangwava, Tinashe B. Mubvirwi Design and Simulation of IoT Systems Using the Cisco Packet Tracer, -ISSN Online: 2161-6825 -ISSN Print: 2161-6817 (дата звернення 12.05.2023)

20. Мікроклімат виробничих приміщень та його вплив на організм працівника: URL https://oppb.com.ua/news/mikroklimat-vyrobnychyh-prymishchen-ta-yogo-vplyv-na-organizm-pracivnyka (дата звернення 24.03.2023)

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

app.py

```
from flask import Flask, render_template, session, request, redirect, send_from_directory, url_for
from datetime import datetime, timedelta
from pymongo import MongoClient, DESCENDING
from flask_mail import Mail, Message
from apscheduler.schedulers.background import BackgroundScheduler
import os
app = Flask(__name__)
app.static_folder = 'static'
app.secret_key = 'your_secret_key_here'
app.config['MAIL_SERVER'] = 'smtp.gmail.com'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USERNAME'] = 'pd12vvvv@gmail.com'
app.config['MAIL_PASSWORD'] = 'kbiqsiwhihyfshgg'
app.config['MAIL_USE_SSL'] = True
mail = Mail(app)
scheduler = BackgroundScheduler()
scheduler.start()
client = MongoClient(
"mongodb+srv://daria:o9MOBaU8XNf12Twc@cluster0.gqaaypy.mongodb.net/?retryWrites=true&w=majority")
try:
    client.admin.command('ping')
    print("Pinged your deployment. You successfully connected to MongoDB!")
except Exception as e:
    print(e)
db = client.iot_data
settings_collection = db.area_data_settings
temperature = db.temperature_data
humidity = db.humidity_data
co2 = db.co2_data
light = db.light_data
users = db.users
@app.before_first_request
def make_session_permanent():
    session.permanent = True
    app.permanent_session_lifetime = timedelta(minutes=5)
@app.route('/static')
def serve_static(path):
    return send_from_directory('static', path)
def get_last_temperature_by_area(area: str):
    return temperature.find_one({'area': area}, sort=[("_id", DESCENDING)])
def get_last_humidity_by_area(area: str):
    return humidity.find_one({'area': area}, sort=[("_id", DESCENDING)])
def get_last_co2_by_area(area: str):
    return co2.find_one({'area': area}, sort=[("_id", DESCENDING)])
def get_last_light_by_area(area: str):
    return light.find_one({'area': area}, sort=[("_id", DESCENDING)])
def send_email():
    with app.app_context():
        msg = Message('Hello', sender='pd12vvvv@gmail.com',
                      recipients=['testaccmail1d@gmail.com'])
        latest_date = {
            'temperature_aa': round(float(get_last_temperature_by_area('AA')['temperature']), 1),
            'temperature_pa': round(float(get_last_temperature_by_area('PA')['temperature']), 1),
```

```
            'temperature_sda': round(float(get_last_temperature_by_area('SDA')['temperature']), 1),
            'temperature_rsa': round(float(get_last_temperature_by_area('RSA')['temperature']), 1),
            'temperature_qa': round(float(get_last_temperature_by_area('QA')['temperature']), 1),
            'humidity_aa': round(float(get_last_humidity_by_area('AA')['humidity_level']), 1),
            'humidity_pa': round(float(get_last_humidity_by_area('PA')['humidity_level']), 1),
            'humidity_sda': round(float(get_last_humidity_by_area('SDA')['humidity_level']), 1),
            'humidity_rsa': round(float(get_last_humidity_by_area('RSA')['humidity_level']), 1),
            'humidity_qa': round(float(get_last_humidity_by_area('QA')['humidity_level']), 1),
            'co2_aa': round(float(get_last_co2_by_area('AA')['co2_value']), 1),
            'co2_pa': round(float(get_last_co2_by_area('PA')['co2_value']), 1),
            'co2_sda': round(float(get_last_co2_by_area('SDA')['co2_value']), 1),
            'co2_rsa': round(float(get_last_co2_by_area('RSA')['co2_value']), 1),
            'co2_qa': round(float(get_last_co2_by_area('QA')['co2_value']), 1),
            'light_aa': round(float(get_last_light_by_area('AA')['light']), 1),
            'light_pa': round(float(get_last_light_by_area('PA')['light']), 1),
            'light_sda': round(float(get_last_light_by_area('SDA')['light']), 1),
            'light_rsa': round(float(get_last_light_by_area('RSA')['light']), 1),
            'light_qa': round(float(get_last_light_by_area('QA')['light']), 1),
            'date': str(get_last_humidity_by_area('AA')['date'].date()),
            'time': str(get_last_humidity_by_area('AA')['date'].time())
        }
        msg.html = render_template('mail.html', latest_date=latest_date)
        print(get_last_humidity_by_area('AA')['date'].time())
        mail.send(msg)
        print('was sent')
@app.route('/profile.html', methods=['GET'])
def profile_page():
    if ('email' in session and users.find_one({'email': session['email']})):
        filter = {'email': session['email']}
        user = users.find_one(filter)
        return render_template('profile.html', user=user)
def dashboard():
    if ('email' in session and users.find_one({'email': session['email']})):
        return render_template('dashboard.html')
@app.route('/aa.html', methods=['GET'])
def aa_page():
    if ('email' in session and users.find_one({'email': session['email']})):
        error_message = None
        filter_area = {"area": "AA"}
        last_co2 = co2.find_one(
            filter_area, sort=[("_id", DESCENDING)])
        last_light = light.find_one(
            filter_area, sort=[("_id", DESCENDING)])
        humidifier_is_on = settings_collection.find_one({'area_name': "AA"})
        fan_is_on = settings_collection.find_one({'area_name': "AA"})
        last_ten_records_humidity = humidity.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        last_ten_records_co2 = co2.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        last_ten_records_light = light.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        last_ten_records_temperature = temperature.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        last_ten_records_humidity_for_date = humidity.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        user = users.find_one({'email': session['email']})
        latest_data = {
            'temperature': round(float(get_last_temperature_by_area('AA')['temperature']), 1),
            'co2': round(float(last_co2['co2_value']), 1),
            'humidity': round(float(get_last_humidity_by_area('AA')['humidity_level']), 1),
            'light': round(float(last_light['light']), 1),
            'humidifier_status': humidifier_is_on['humidifier_is_on'],
            'fan_status': fan_is_on['fan'],
```

```
            'chart_date': [record['date'] for record in last_ten_records_humidity_for_date],
            'chart_light': [record['light'] for record in last_ten_records_light],
            'chart_humidity': [round(float(x['humidity_level']), 1) for x in last_ten_records_humidity],
            'chart_temperature': [record['temperature'] for record in last_ten_records_temperature],
            'chart_co2': [record['co2_value'] for record in last_ten_records_co2],
            'name': f"{user['first_name']} {user['last_name']}",
            'work': user['work']
        }
        return render_template('aa.html',  latest_data=latest_data, error_message=error_message)
    else:
        return render_template('login.html')

@app.route('/pa.html', methods=['GET'])
def pa_page():
    if ('email' in session and users.find_one({'email': session['email']})):
        filter_area = {"area": "PA"}
        last_co2 = co2.find_one(
            filter_area, sort=[("_id", DESCENDING)])
        last_light = light.find_one(
            filter_area, sort=[("_id", DESCENDING)])
        humidifier_is_on = settings_collection.find_one({'area_name': "PA"})
        fan_is_on = settings_collection.find_one({'area_name': "PA"})
        last_ten_records_humidity = humidity.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        last_ten_records_co2 = co2.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        last_ten_records_light = light.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        last_ten_records_temperature = temperature.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        last_ten_records_humidity_for_date = humidity.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        user = users.find_one({'email': session['email']})
        latest_data = {
            'temperature': round(float(get_last_temperature_by_area('PA')['temperature']), 1),
            'co2': round(float(last_co2['co2_value']), 1),
            'humidity': round(float(get_last_humidity_by_area('PA')['humidity_level']), 1),
            'light': round(float(last_light['light']), 1),
            'humidifier_status': humidifier_is_on['humidifier_is_on'],
            'fan_status': fan_is_on['fan'],
            'chart_date': [record['date'] for record in last_ten_records_humidity_for_date],
            'chart_light': [record['light'] for record in last_ten_records_light],
            'chart_humidity': [round(float(x['humidity_level']), 1) for x in last_ten_records_humidity],
            'chart_temperature': [record['temperature'] for record in last_ten_records_temperature],
            'chart_co2': [record['co2_value'] for record in last_ten_records_co2],
            'name': f"{user['first_name']} {user['last_name']}",
            'work': user['work']
        }
        return render_template('pa.html',  latest_data=latest_data)
    else:
        return render_template('login.html')
@app.route('/qa.html', methods=['GET'])
def qa_page():
    if ('email' in session and users.find_one({'email': session['email']})):
        user = users.find_one({'email': session['email']})
        filter_area = {"area": "QA"}
        last_co2 = co2.find_one(
            filter_area, sort=[("_id", DESCENDING)])
        last_light = light.find_one(
            filter_area, sort=[("_id", DESCENDING)])
        humidifier_is_on = settings_collection.find_one({'area_name': "QA"})
        fan_is_on = settings_collection.find_one({'area_name': "QA"})
        last_ten_records_humidity = humidity.find(
```

```
            filter_area).sort('_id', DESCENDING).limit(10)
        last_ten_records_co2 = co2.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        last_ten_records_light = light.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        last_ten_records_temperature = temperature.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        last_ten_records_humidity_for_date = humidity.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        latest_data = {
            'temperature': round(float(get_last_temperature_by_area('QA')['temperature']), 1),
            'co2': round(float(last_co2['co2_value']), 1),
            'humidity': round(float(get_last_humidity_by_area('QA')['humidity_level']), 1),
            'light': round(float(last_light['light']), 1),
            'humidifier_status': humidifier_is_on['humidifier_is_on'],
            'fan_status': fan_is_on['fan'],
            'chart_date': [record['date'] for record in last_ten_records_humidity_for_date],
            'chart_light': [record['light'] for record in last_ten_records_light],
            'chart_humidity': [round(float(x['humidity_level']), 1) for x in last_ten_records_humidity],
            'chart_temperature': [record['temperature'] for record in last_ten_records_temperature],
            'chart_co2': [record['co2_value'] for record in last_ten_records_co2],
            'name': f"{user['first_name']} {user['last_name']}",
            'work': user['work']
        }
        print(latest_data)
        return render_template('qa.html',  latest_data=latest_data)
    else:
        return render_template('login.html')
@app.route('/rsa.html', methods=['GET'])
def rsa_page():
    if ('email' in session and users.find_one({'email': session['email']})):
        user = users.find_one({'email': session['email']})
        filter_area = {"area": "RSA"}
        last_co2 = co2.find_one(
            filter_area, sort=[("_id", DESCENDING)])
        last_light = light.find_one(
            filter_area, sort=[("_id", DESCENDING)])
        humidifier_is_on = settings_collection.find_one({'area_name': "RSA"})
        fan_is_on = settings_collection.find_one({'area_name': "RSA"})
        last_ten_records_humidity = humidity.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        last_ten_records_co2 = co2.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        last_ten_records_light = light.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        last_ten_records_temperature = temperature.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        last_ten_records_humidity_for_date = humidity.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        latest_data = {
            'temperature': round(float(get_last_temperature_by_area('RSA')['temperature']), 1),
            'co2': round(float(last_co2['co2_value']), 1),
            'humidity': round(float(get_last_humidity_by_area('RSA')['humidity_level']), 1),
            'light': round(float(last_light['light']), 1),
            'humidifier_status': humidifier_is_on['humidifier_is_on'],
            'fan_status': fan_is_on['fan'],
            'chart_date': [record['date'] for record in last_ten_records_humidity_for_date],
            'chart_light': [record['light'] for record in last_ten_records_light],
            'chart_humidity': [round(float(x['humidity_level']), 1) for x in last_ten_records_humidity],
            'chart_temperature': [record['temperature'] for record in last_ten_records_temperature],
            'chart_co2': [record['co2_value'] for record in last_ten_records_co2],
            'name': f"{user['first_name']} {user['last_name']}",
            'work': user['work']
```

```
        }
        print(latest_data)
        return render_template('rsa.html',  latest_data=latest_data)
    else:
        return render_template('login.html')
@app.route('/sda.html', methods=['GET'])
def sda_page():
    if ('email' in session and users.find_one({'email': session['email']})):
        user = users.find_one({'email': session['email']})
        filter_area = {"area": "SDA"}
        last_co2 = co2.find_one(
            filter_area, sort=[("_id", DESCENDING)])
        last_light = light.find_one(
            filter_area, sort=[("_id", DESCENDING)])
        humidifier_is_on = settings_collection.find_one({'area_name': "SDA"})
        fan_is_on = settings_collection.find_one({'area_name': "SDA"})
        last_ten_records_humidity = humidity.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        last_ten_records_co2 = co2.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        last_ten_records_light = light.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        last_ten_records_temperature = temperature.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        last_ten_records_humidity_for_date = humidity.find(
            filter_area).sort('_id', DESCENDING).limit(10)
        latest_data = {
            'temperature': round(float(get_last_temperature_by_area('SDA')['temperature']), 1),
            'co2': round(float(last_co2['co2_value']), 1),
            'humidity': round(float(get_last_humidity_by_area('SDA')['humidity_level']), 1),
            'light': round(float(last_light['light']), 1),
            'humidifier_status': humidifier_is_on['humidifier_is_on'],
            'fan_status': fan_is_on['fan'],
            'chart_date': [record['date'] for record in last_ten_records_humidity_for_date],
            'chart_light': [record['light'] for record in last_ten_records_light],
            'chart_humidity': [round(float(x['humidity_level']), 1) for x in last_ten_records_humidity],
            'chart_temperature': [record['temperature'] for record in last_ten_records_temperature],
            'chart_co2': [record['co2_value'] for record in last_ten_records_co2],
            'name': f"{user['first_name']} {user['last_name']}",
            'work': user['work']
        }
        print(latest_data)
        return render_template('sda.html',  latest_data=latest_data)
    else:
        return render_template('login.html')
@app.route('/update_profile', methods=['POST'])
def update_profile():
    if ('email' in session and users.find_one({'email': session['email']})):
        updated_data = request.form.to_dict()

        print(updated_data)
        set_new_user_information = {
            '$set': {
                'first_name': updated_data['name'],
                'last_name': updated_data['last_name'],
                'work': updated_data['position']
            }
        }
        users.update_one({"email": session['email']}, set_new_user_information)
        return redirect('/profile.html')
    else:
        return render_template('login.html')
@app.route('/')
```

```
def start():
    if ('email' in session and users.find_one({'email': session['email']})):
        user = users.find_one({'email': session['email']})
        latest_data = {
            'name': f"{user['first_name']} {user['last_name']}",
            'work': user['work']
        }
        return render_template('dashboard.html', latest_data=latest_data)
    else:
        return render_template('login.html')
@app.route('/login', methods=['POST'])
def login():
    error = None
    email = request.form['email']
    password = request.form['password']
    user = users.find_one({'email': email})
    print(f"User email: {user}")
    if user:
        if user['password'] == password:
            session['email'] = email
            # Successful login
            return redirect(url_for('dashboard'))
        elif user['password'] != password:
            error = 'Wrong password provided.'
        else:
            error = 'Some error. User exists in system.'
    elif user is None:
        error = "No such user in our system. Please check your credintials."
    else:
        # Failed login
        error = "Error"
    return render_template('login.html', error=error)
@app.route('/dashboard')
def dashboard():
    if ('email' in session and users.find_one({'email': session['email']})):
        user = users.find_one({'email': session['email']})
        latest_data = {
            'name': f"{user['first_name']} {user['last_name']}",
            'work': user['work']
        }
        return render_template('dashboard.html', latest_data=latest_data)
    else:
        return render_template("login.html")
@app.route('/get_auto_value_toggle', methods=['POST'])
def get_auto_value_toggle():
    if ('email' in session and users.find_one({'email': session['email']})):
        referer_url = request.headers.get('referer')
        temperature_auto = None
        cool_is_on = None
        heat_is_on = None
        updates_values_for_auto = None
        print(referer_url)
        state = request.json['state']
        name = request.json['toggle_name']
        # Do something with the toggle button state
        print(state, name)
        if state == 'checked' and name == 'auto':
            temperature_auto = True
            cool_is_on = False
            heat_is_on = False
            updates_values_for_auto = {
                '$set': {
                    'temperature_auto': temperature_auto,
```

```
                'cool_is_on': cool_is_on,
                'heat_is_on': heat_is_on
            }
        }
    elif state == 'checked' and name == 'heat':
        temperature_auto = False
        cool_is_on = False
        heat_is_on = True
        updates_values_for_auto = {
            '$set': {
                'temperature_auto': temperature_auto,
                'cool_is_on': cool_is_on,
                'heat_is_on': heat_is_on
            }
        }
    elif state == 'checked' and name == 'cool':
        temperature_auto = False
        cool_is_on = True
        heat_is_on = False
        updates_values_for_auto = {
            '$set': {
                'temperature_auto': temperature_auto,
                'cool_is_on': cool_is_on,
                'heat_is_on': heat_is_on
            }
        }
    if 'aa.html' in referer_url:
        print(updates_values_for_auto)
        area_filter = {"area_name": "AA"}
        settings_collection.update_one(
            area_filter, updates_values_for_auto)
        return redirect("/aa.html")
    elif 'pa.html' in referer_url:
        print(updates_values_for_auto)
        area_filter = {"area_name": "PA"}
        settings_collection.update_one(
            area_filter, updates_values_for_auto)
        return redirect("/pa.html")
    elif 'qa.html' in referer_url:
        print(updates_values_for_auto)
        area_filter = {"area_name": "QA"}
        settings_collection.update_one(
            area_filter, updates_values_for_auto)
        return redirect("/qa.html")
    elif 'rsa.html' in referer_url:
        print(updates_values_for_auto)
        area_filter = {"area_name": "RSA"}
        settings_collection.update_one(
            area_filter, updates_values_for_auto)
        return redirect("/rsa.html")
    elif 'sda.html' in referer_url:
        print(updates_values_for_auto)
        area_filter = {"area_name": "SDA"}
        settings_collection.update_one(
            area_filter, updates_values_for_auto)
        return redirect("/sda.html")
  else:
    return render_template("login.html")
@app.route('/get_temperature_update', methods=['POST'])
def get_temperature_update():
  if ('email' in session and users.find_one({'email': session['email']})):
    referer_url = request.headers.get('referer')
    min_temp = request.form['min_temp']
```

```
        max_temp = request.form['max_temp']
        update_temprerature_for_selected_area = {
            '$set': {
                'temperature_min': float(min_temp),
                'temperature_max': float(max_temp)
            }
        }
        if 'aa.html' in referer_url:
            area_filter = {"area_name": "AA"}
            settings_collection.update_one(
                area_filter, update_temprerature_for_selected_area)
            return redirect("aa.html")
        elif 'pa.html' in referer_url:
            area_filter = {"area_name": "PA"}
            settings_collection.update_one(
                area_filter, update_temprerature_for_selected_area)
            return redirect('pa.html')
        elif 'qa.html' in referer_url:
            area_filter = {"area_name": "QA"}
            settings_collection.update_one(
                area_filter, update_temprerature_for_selected_area)
            return redirect('qa.html')
        elif 'rsa.html' in referer_url:
            area_filter = {"area_name": "RSA"}
            settings_collection.update_one(
                area_filter, update_temprerature_for_selected_area)
            return redirect('rsa.html')
        elif 'sda.html' in referer_url:
            area_filter = {"area_name": "SDA"}
            settings_collection.update_one(
                area_filter, update_temprerature_for_selected_area)
            return redirect('sda.html')
    else:
        return render_template("login.html")
@app.route('/get_humidity_update', methods=['POST'])
def get_humidity_update():
    if ('email' in session and users.find_one({'email': session['email']})):
        referer_url = request.headers.get('referer')
        min_hum = request.form['min_hum']
        max_hum = request.form['max_hum']
        update_humidity_for_selected_area = {
            '$set': {
                'humidity_min': float(min_hum),
                'humidity_max': float(max_hum)
            }
        }
        if 'aa.html' in referer_url:
            area_filter = {"area_name": "AA"}
            settings_collection.update_one(
                area_filter, update_humidity_for_selected_area)
            return redirect("aa.html")
        elif 'pa.html' in referer_url:
            area_filter = {"area_name": "PA"}
            settings_collection.update_one(
                area_filter, update_humidity_for_selected_area)
            return redirect('pa.html')
        elif 'qa.html' in referer_url:
            area_filter = {"area_name": "QA"}
            settings_collection.update_one(
                area_filter, update_humidity_for_selected_area)
            return redirect('qa.html')
        elif 'rsa.html' in referer_url:
            area_filter = {"area_name": "RSA"}
```

```
            settings_collection.update_one(
                area_filter, update_humidity_for_selected_area)
            return redirect('rsa.html')
        elif 'sda.html' in referer_url:
            area_filter = {"area_name": "SDA"}
            settings_collection.update_one(
                area_filter, update_humidity_for_selected_area)
            return redirect('sda.html')
    else:
        return render_template("login.html")
@app.route('/get_co2_update', methods=['POST'])
def get_co2_update():
    if ('email' in session and users.find_one({'email': session['email']})):
        referer_url = request.headers.get('referer')
        min_co2 = request.form['min_co2']
        max_co2 = request.form['max_co2']
        # return redirect(os.path.basename(referer_url))
        print(os.path.basename(referer_url))
        update_co2_for_selected_area = {
            '$set': {
                'co2_level_min': float(min_co2),
                'co2_level_max': float(max_co2)
            }
        }
        if 'aa.html' in referer_url:
            area_filter = {"area_name": "AA"}
            settings_collection.update_one(
                area_filter, update_co2_for_selected_area)
            return redirect("aa.html")
        elif 'pa.html' in referer_url:
            area_filter = {"area_name": "PA"}
            settings_collection.update_one(
                area_filter, update_co2_for_selected_area)
            return redirect('pa.html')
        elif 'qa.html' in referer_url:
            area_filter = {"area_name": "QA"}
            settings_collection.update_one(
                area_filter, update_co2_for_selected_area)
            return redirect('qa.html')
        elif 'sda.html' in referer_url:
            area_filter = {"area_name": "SDA"}
            settings_collection.update_one(
                area_filter, update_co2_for_selected_area)
            return redirect('sda.html')
        elif 'rsa.html' in referer_url:
            area_filter = {"area_name": "RSA"}
            settings_collection.update_one(
                area_filter, update_co2_for_selected_area)
            return redirect('rsa.html')
    else:
        return render_template("login.html")
@app.route('/logout')
def logout():
    session.pop('email', None)
    return render_template("login.html")
if __name__ == "__main__":
    scheduler.add_job(send_email, 'interval', minutes=5)
    app.run()

    Main.py
    from room import Room
    from humidity import Humidifier, HumiditySensor
    from temperature import TemperatureSensor, AC
```

```
from co2 import Fan, CO2Sensor
from light import LightAutomation, LightSensor
from time import sleep
from datetime import datetime, timedelta
from pymongo import MongoClient, DESCENDING

# Connect to MongoDB
client = MongoClient(
    "mongodb+srv://daria:o9MOBaU8XNf12Twc@cluster0.gqaaypy.mongodb.net/?retryWrites=true&w=majority")
db = client.iot_data
collection_temperature = db.temperature_data
collection_humidity = db.humidity_data
collection_area_settings = db.area_data_settings
collection_co2 = db.co2_data
collection_light = db.light_data

def simulate():

    while True:
        light_sensor = LightSensor()
        humidity_sensor = HumiditySensor()
        temperature_sensor = TemperatureSensor()
        co2_sensor = CO2Sensor()
        global humidity, temperature, co2, light
        # for AA room
        AA_MAX_TENPERATURE = collection_area_settings.find_one(
            filter={"area_name": "AA"})['temperature_max']
        AA_MIN_TENPERATURE = collection_area_settings.find_one(
            filter={"area_name": "AA"})['temperature_min']
        AA_MIN_HUMIDITY = collection_area_settings.find_one(
            filter={"area_name": "AA"})['humidity_min']
        AA_MAX_HUMIDITY = collection_area_settings.find_one(
            filter={"area_name": "AA"})['humidity_max']
        AA_MIN_CO2 = collection_area_settings.find_one(
            filter={"area_name": "AA"})['co2_level_min']
        AA_MAX_CO2 = collection_area_settings.find_one(
            filter={"area_name": "AA"})['co2_level_max']
        AA_AUTO_TEMP_STATUS = collection_area_settings.find_one(
            filter={"area_name": "AA"})['temperature_auto']
        AA_HEATER_STATUS = collection_area_settings.find_one(
            filter={"area_name": "AA"})['heat_is_on']
        AA_COOLER_STATUS = collection_area_settings.find_one(
            filter={"area_name": "AA"})['cool_is_on']
        AA_LIGHT_USAGE = collection_area_settings.find_one(
            filter={"area_name": "AA"})['light_usage']
        # for QA room
        QA_MAX_TENPERATURE = collection_area_settings.find_one(
            filter={"area_name": "QA"})['temperature_max']
        QA_MIN_TENPERATURE = collection_area_settings.find_one(
            filter={"area_name": "QA"})['temperature_min']
        QA_MIN_HUMIDITY = collection_area_settings.find_one(
            filter={"area_name": "QA"})['humidity_min']
        QA_MAX_HUMIDITY = collection_area_settings.find_one(
            filter={"area_name": "QA"})['humidity_max']
        QA_MIN_CO2 = collection_area_settings.find_one(
            filter={"area_name": "QA"})['co2_level_min']
        QA_MAX_CO2 = collection_area_settings.find_one(
            filter={"area_name": "QA"})['co2_level_max']
        QA_AUTO_TEMP_STATUS = collection_area_settings.find_one(
            filter={"area_name": "QA"})['temperature_auto']
        QA_HEATER_STATUS = collection_area_settings.find_one(
            filter={"area_name": "QA"})['heat_is_on']
        QA_COOLER_STATUS = collection_area_settings.find_one(
```

```
    filter={"area_name": "QA"})['cool_is_on']
QA_LIGHT_USAGE = collection_area_settings.find_one(
    filter={"area_name": "QA"})['light_usage']
# for SDA room
SDA_MAX_TENPERATURE = collection_area_settings.find_one(
    filter={"area_name": "SDA"})['temperature_max']
SDA_MIN_TENPERATURE = collection_area_settings.find_one(
    filter={"area_name": "SDA"})['temperature_min']
SDA_MIN_HUMIDITY = collection_area_settings.find_one(
    filter={"area_name": "SDA"})['humidity_min']
SDA_MAX_HUMIDITY = collection_area_settings.find_one(
    filter={"area_name": "SDA"})['humidity_max']
SDA_MIN_CO2 = collection_area_settings.find_one(
    filter={"area_name": "SDA"})['co2_level_min']
SDA_MAX_CO2 = collection_area_settings.find_one(
    filter={"area_name": "SDA"})['co2_level_max']
SDA_AUTO_TEMP_STATUS = collection_area_settings.find_one(
    filter={"area_name": "SDA"})['temperature_auto']
SDA_HEATER_STATUS = collection_area_settings.find_one(
    filter={"area_name": "SDA"})['heat_is_on']
SDA_COOLER_STATUS = collection_area_settings.find_one(
    filter={"area_name": "SDA"})['cool_is_on']
SDA_LIGHT_USAGE = collection_area_settings.find_one(
    filter={"area_name": "SDA"})['light_usage']
# for PA room
PA_MAX_TENPERATURE = collection_area_settings.find_one(
    filter={"area_name": "PA"})['temperature_max']
PA_MIN_TENPERATURE = collection_area_settings.find_one(
    filter={"area_name": "PA"})['temperature_min']
PA_MIN_HUMIDITY = collection_area_settings.find_one(
    filter={"area_name": "PA"})['humidity_min']
PA_MAX_HUMIDITY = collection_area_settings.find_one(
    filter={"area_name": "PA"})['humidity_max']
PA_MIN_CO2 = collection_area_settings.find_one(
    filter={"area_name": "PA"})['co2_level_min']
PA_MAX_CO2 = collection_area_settings.find_one(
    filter={"area_name": "PA"})['co2_level_max']
PA_AUTO_TEMP_STATUS = collection_area_settings.find_one(
    filter={"area_name": "PA"})['temperature_auto']
PA_HEATER_STATUS = collection_area_settings.find_one(
    filter={"area_name": "PA"})['heat_is_on']
PA_COOLER_STATUS = collection_area_settings.find_one(
    filter={"area_name": "PA"})['cool_is_on']
PA_LIGHT_USAGE = collection_area_settings.find_one(
    filter={"area_name": "PA"})['light_usage']
# for RSA room
RSA_MAX_TENPERATURE = collection_area_settings.find_one(
    filter={"area_name": "RSA"})['temperature_max']
RSA_MIN_TENPERATURE = collection_area_settings.find_one(
    filter={"area_name": "RSA"})['temperature_min']
RSA_MIN_HUMIDITY = collection_area_settings.find_one(
    filter={"area_name": "RSA"})['humidity_min']
RSA_MAX_HUMIDITY = collection_area_settings.find_one(
    filter={"area_name": "RSA"})['humidity_max']
RSA_MIN_CO2 = collection_area_settings.find_one(
    filter={"area_name": "RSA"})['co2_level_min']
RSA_MAX_CO2 = collection_area_settings.find_one(
    filter={"area_name": "RSA"})['co2_level_max']
RSA_AUTO_TEMP_STATUS = collection_area_settings.find_one(
    filter={"area_name": "RSA"})['temperature_auto']
RSA_HEATER_STATUS = collection_area_settings.find_one(
    filter={"area_name": "RSA"})['heat_is_on']
RSA_COOLER_STATUS = collection_area_settings.find_one(
```

```
        filter={"area_name": "RSA"})['cool_is_on']
    RSA_LIGHT_USAGE = collection_area_settings.find_one(
        filter={"area_name": "RSA"})['light_usage']
    rooms = [
            Room('QA', QA_MAX_TENPERATURE, QA_MIN_TENPERATURE, QA_MIN_HUMIDITY,
QA_MAX_HUMIDITY, QA_MIN_CO2, QA_MAX_CO2, QA_AUTO_TEMP_STATUS,
            QA_HEATER_STATUS, QA_COOLER_STATUS, QA_LIGHT_USAGE),
            Room('SDA', SDA_MAX_TENPERATURE, SDA_MIN_TENPERATURE, SDA_MIN_HUMIDITY,
SDA_MAX_HUMIDITY, SDA_MIN_CO2, SDA_MAX_CO2, SDA_AUTO_TEMP_STATUS,
            SDA_HEATER_STATUS, SDA_COOLER_STATUS, SDA_LIGHT_USAGE),
            Room('PA', PA_MAX_TENPERATURE, PA_MIN_TENPERATURE, PA_MIN_HUMIDITY,
PA_MAX_HUMIDITY, PA_MIN_CO2, PA_MAX_CO2, PA_AUTO_TEMP_STATUS,
            PA_HEATER_STATUS, PA_COOLER_STATUS, PA_LIGHT_USAGE),
            Room('AA', AA_MAX_TENPERATURE, AA_MIN_TENPERATURE, AA_MIN_HUMIDITY,
AA_MAX_HUMIDITY, AA_MIN_CO2, AA_MAX_CO2, AA_AUTO_TEMP_STATUS,
            AA_HEATER_STATUS, AA_COOLER_STATUS, AA_LIGHT_USAGE),
            Room('RSA', RSA_MAX_TENPERATURE, RSA_MIN_TENPERATURE, RSA_MIN_HUMIDITY,
RSA_MAX_HUMIDITY, RSA_MIN_CO2, RSA_MAX_CO2, RSA_AUTO_TEMP_STATUS,
            RSA_HEATER_STATUS, RSA_COOLER_STATUS, RSA_LIGHT_USAGE)
    ]
    for room in rooms:
        first_datetime = 0
        hour_delta = timedelta(hours=3)
        if collection_humidity.find_one(sort=[("_id", DESCENDING)])['date'] is None:
            first_datetime = datetime(2023, 4, 1, 0, 0, 0)
        else:
            first_datetime = collection_humidity.find_one(
                sort=[("_id", DESCENDING)])['date']

        if room.name == "AA":
            humidity = humidity_sensor.get_last_humidity_data(room.name)
            temperature = temperature_sensor.get_last_temperature_data(
                room.name)
            co2 = co2_sensor.get_last_co2_data(room.name)
            light = light_sensor.get_last_light_data(room.name)

        elif room.name == "QA":
            humidity = humidity_sensor.get_last_humidity_data(room.name)
            temperature = temperature_sensor.get_last_temperature_data(
                room.name)
            co2 = co2_sensor.get_last_co2_data(room.name)
            light = light_sensor.get_last_light_data(room.name)
        elif room.name == "PA":
            humidity = humidity_sensor.get_last_humidity_data(room.name)
            temperature = temperature_sensor.get_last_temperature_data(
                room.name)
            co2 = co2_sensor.get_last_co2_data(room.name)
            light = light_sensor.get_last_light_data(room.name)
        elif room.name == "SDA":
            humidity = humidity_sensor.get_last_humidity_data(room.name)
            temperature = temperature_sensor.get_last_temperature_data(
                room.name)
            co2 = co2_sensor.get_last_co2_data(room.name)
            light = light_sensor.get_last_light_data(room.name)
        elif room.name == "RSA":
            humidity = humidity_sensor.get_last_humidity_data(room.name)
            temperature = temperature_sensor.get_last_temperature_data(
                room.name)
            co2 = co2_sensor.get_last_co2_data(room.name)
            light = light_sensor.get_last_light_data(room.name)

        humidifier = Humidifier(humidity, room)
        humidity_level = humidifier.detect_humidity()
```

```
            ac = AC(temperature, room, first_datetime)
            temperature_value = ac.temp_detect()
            fan = Fan(co2, room)
            co2_value = fan.co2_detector()
            light_automation = LightAutomation(light, room, first_datetime)
            light_value = light_automation.detect_light()
            light_data = {}
            updated_datetime = first_datetime + hour_delta
            if light_value is None:
                print("Light was off")
            else:
                light_data = {
                    "area": light_value[2],
                    "date": updated_datetime,
                    "light": light_value[1],
                    "cost": light_value[0]
                }
                # energy_cost, current_light, room name
                print(
                    f"Light: {light_value[0], light_value[1], light_value[2]}")
                collection_light.insert_one(light_data)

            humidity_data = {
                'area': room.name,
                'date': updated_datetime,
                'humidity_level': humidity_level[0]
            }
            co2_data = {
                'area': room.name,
                'date': updated_datetime,
                'co2_value': co2_value[0]
            }

            temperature_data = {
                'area': room.name,
                'temperature': temperature_value[0],
                'date': updated_datetime,
            }
            if humidity_level is None:
                print(f"Humidity level is: {humidity_level}")
                print(f"Temp data is: {temperature_data}")
            else:
                print(humidity_data)
                print(temperature_data)
                print(co2_data)
                print(light_data)
                # collection_light.insert_one(light_data)
                collection_humidity.insert_one(humidity_data)
                collection_temperature.insert_one(temperature_data)
                collection_co2.insert_one(co2_data)

if __name__ == "__main__":
    simulate()
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

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

ВІДГУК КЕРІВНИКА КВАЛІФІКАЦІЙНОЇ РОБОТИ

РЕЦЕНЗІЯ