ДОДАТОК А

Звіт результатів перевірки на унікальність тексту

StrikePlagiarism.com

Дата звіту 6/7/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_Б_ПІ_Пр_ПЗПІ_21_10_Ільєнко_Б_А_скорочений
Автор
Ільєнко Богдан Анатолійович
Науковий керівник / Експерт
Колесников Д.О./Нечволод В.Ю.
підрозділ
каф. ПІ

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

1.09% 1.09% КП 1

0.27% 0.27% КЦ

25	4418	36479
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	¶	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	5

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://duikt.edu.ua/repozitorii/ipz/2024/%D0%9F%D0%94-42/%D0%9F%D0%94-42%20%D0%9C%D0%B0%D0%BD%D1%83%D1%88%D0%BA%D1%96%D0%BD%20%D0%90.%D0%84..pdf	17 0.38 %
2	https://openarchive.nure.ua/bitstreams/844269c9-b132-4fa6-91d7-3d0bfcaee5e4/download	12 0.27 %
3	https://openarchive.nure.ua/bitstreams/dcb865f6-766f-4683-a4bb-5cb937468862/download	11 0.25 %
4	https://openarchive.nure.ua/bitstreams/dcb865f6-766f-4683-a4bb-5cb937468862/download	8 0.18 %

ДОДАТОК Б

Слайди презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

NURE ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Програмна система для автоматизованого управління освітленням для вирощування рослин. Фронт-енд

ПІБ, група Ільєнко Богдан Анатолійович ПЗПІ-21-10

Керівник: доц. кафедри ПІ Дмитро КОЛЕСНИКОВ

SE software engineering

3 червня 2025

Мета роботи

Розробити клієнтську частину програмної системи для керування освітленням рослин, яка надає повнофункціональний веб-інтерфейс із підтримкою адаптивного дизайну, зручної навігації та взаємодії з серверним API. Система має забезпечувати ефективну роботу для всіх типів користувачів — адміністратора, технічного персоналу та звичайного користувача — з урахуванням їхніх прав доступу.

Передбачено використання сесійної авторизації для безпечної роботи із захищеними маршрутами, а також механізмів отримання та оновлення даних у режимі реального часу з сенсорів, підключених до IoT-пристроїв. Фронтенд реалізує обробку помилок, кешування відповідей, перевірку автентифікації, а також можливість керування сутностями системи: рослинами, сенсорами, освітлювальними пристроями та користувачами.

SE software engineering

2

Аналіз проблеми

– В агросекторі відсутні прості у використанні інтерфейси для моніторингу стану рослин.

– Існуючі рішення складні або орієнтовані на великі господарства.

– Потрібна інтуїтивна фронтенд-система з ролями, захистом даних і підтримкою IoT.

– Використання сучасного стеку (React, TypeScript, RTK Query) дозволяє реалізувати масштабований SPA-додаток.

SE software engineering

3

Аналіз існуючих рішень

Критерій	Fluence	Heliospectra	Lumigrow
Веб-інтерфейс	+	+	+/-
Гнучка рольова модель	+/-	-	-
Взаємодія з сенсорами в реальному часі	+	+	+
Імпорт/експорт даних (CSV/JSON)	-	-	+/-
Можливість масштабування	+	+	+
Інтеграція з IoT	+	+/-	+/-
Простота використання UI	+/-	+	-
Підтримка авторизації через сесії	-	-	-

Опис системи

Більшість сучасних систем автоматизованого управління в агросекторі зосереджені виключно на апаратному забезпеченні або серверній логіці, залишаючи користувацький інтерфейс недостатньо опрацьованим. Часто такі інтерфейси є складними у використанні, неадаптованими до ролей користувачів (адміністратор, технік, оператор) і не підтримують інтерактивне керування в реальному часі.

Крім того, у багатьох рішеннях відсутній єдиний веб-застосунок, через що взаємодія з системою потребує спеціалізованого програмного забезпечення або ручної обробки даних. Це значно ускладнює моніторинг стану пристроїв, отримання сенсорних показників та оперативне налаштування параметрів освітлення.

Така ситуація створює бар'єри для впровадження технологій у невеликих господарствах, де потрібна проста, зручна та доступна у використанні система управління з чітким інтерфейсом.

Постановка задачі

У результаті реалізації очікується створення сучасного клієнтського застосунку з використанням React і TypeScript, який забезпечуватиме зручну взаємодію користувача із системою.

Передбачається:

– підтримка автентифікації через сесії та авторизації з урахуванням ролей (адміністратор, технік, користувач);

– захищені маршрути та обмеження доступу до сторінок на основі прав користувача;

– інтеграція з серверним API через RTK Query для зручної роботи з даними;

– реалізація основного функціоналу: перегляд, додавання, редагування і видалення рослин, сенсорів і освітлювальних пристроїв;

– створення зручного і адаптивного інтерфейсу з використанням Ant Design.

Вибір технологій розробки

Для реалізації клієнтської частини було використано наступний стек технологій:

- React — для створення інтерфейсу користувача;
- TypeScript — для статичної типізації й підвищення надійності коду;
- RTK Query (Redux Toolkit) — для організації запитів до API та зберігання глобального стану;
- React Router — для маршрутизації між сторінками;
- Ant Design — для побудови сучасного й адаптивного UI;
- Vite — як збирач проєкту для швидкої розробки.

Архітектура створенного програмного забезпечення

Система побудована за принципами клієнт-серверної архітектури з інтеграцією IoT через протокол MQTT:

Клієнтська частина (React): Відповідає за відображення інтерфейсу, авторизацію через сесії, маршрутизацію (React Router) та запити до API через RTK Query.

Серверна частина (NestJS): Реалізує обробку запитів, бізнес-логіку, роботу з базою даних PostgreSQL, управління сесіями через Redis, а також прийом показників із сенсорів через MQTT.

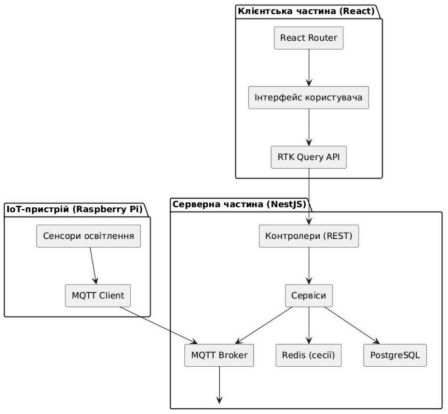
MQTT: Протокол, що забезпечує легку та надійну передачу даних між IoT-пристроєм (Raspberry Pi) та сервером.

PostgreSQL: Основна СУБД для зберігання інформації про користувачів, рослини, сенсори та зчитування.

Redis: Використовується для збереження сесій користувачів у безпечному вигляді.

Архітектура створенного програмного забезпечення (діаграма)

Ця діаграма демонструє загальну архітектуру програмної системи, включаючи взаємодію клієнтської частини, серверу, бази даних, сесій і IoT-пристрою через протокол MQTT. Вона відображає ключові компоненти та спосіб обміну даними між ними.



Опис програмного забезпечення, що було використано у дослідженні

У процесі розробки клієнтської частини застосовувалися сучасні технології та інструменти. Основною мовою програмування став TypeScript, а фреймворком для побудови інтерфейсу — React. Для управління станом і виконання запитів до API було використано Redux Toolkit Query (RTK Query). Для побудови сучасного та зручного інтерфейсу обрано Ant Design. Розробка здійснювалась у середовищі Visual Studio Code із застосуванням системи керування версіями Git.

Дизайн системи

Для проєктування клієнтської частини системи було обрано компонентний підхід, який дозволяє ефективно розділяти відповідальність між окремими частинами інтерфейсу.

Розробка здійснювалась поетапно: спочатку було налаштовано маршрутизацію через React Router, далі — реалізовано базову структуру сторінок, інтеграцію з API через RTK Query та механізми автентифікації через сесії (httpOnly cookies).

Для UI обрано Ant Design, що забезпечило швидку реалізацію форм, таблиць, повідомлень і модальних вікон.

Обробку стану та асинхронних запитів реалізовано через Redux Toolkit Query, що дозволило уникнути зайвого дублювання логіки та забезпечити кешування результатів.

Приклад реалізації

Даний фрагмент коду відповідає за налаштування сесій у серверній частині застосунку на базі NestJS. Сесії зберігаються у Redis за допомогою бібліотеки connect-redis, що забезпечує швидкий доступ та масштабованість. Після входу користувача сервер створює сесію, а браузеру надсилається cookie з ідентифікатором sid. Сесія є захищеною (httpOnly) і має строк дії 7 днів.

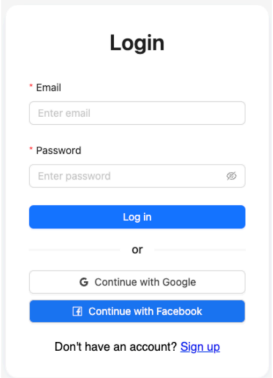
```
import { RedisStore } from "connect-redis"
import Redis from 'ioredis';
import * as session from 'express-session';
import { INestApplication } from "@nestjs/common";


export function configureSession(app: INestApplication) {
    const redisClient = new Redis({ host: 'localhost', port: 6379 });

    app.use(
        session({
            name: 'sid',
            store: new RedisStore({ client: redisClient, disableTouch: true }),
            secret: 'mysecretkey',
            resave: false,
            saveUninitialized: false,
            cookie: {
                httpOnly: true,
                maxAge: 1000 * 60 * 60 * 24 * 7, // 7 дней
                sameSite: 'lax',
            },
        }),
    );
}
```

Інтерфейс користувача (сторінка авторизації)

На початковій сторінці користувача зустрічає форма входу, яка дозволяє авторизуватися через email та пароль. Також реалізовано можливість входу через Google або Facebook. Для нових користувачів передбачене посилання на реєстрацію.



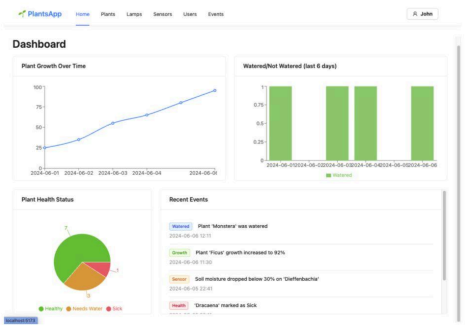
Інтерфейс користувача (головна сторінка)

Цей фрагмент демонструє головну сторінку (Dashboard) клієнтської частини системи PlantsApp, яка надає огляд стану рослин та останніх подій.

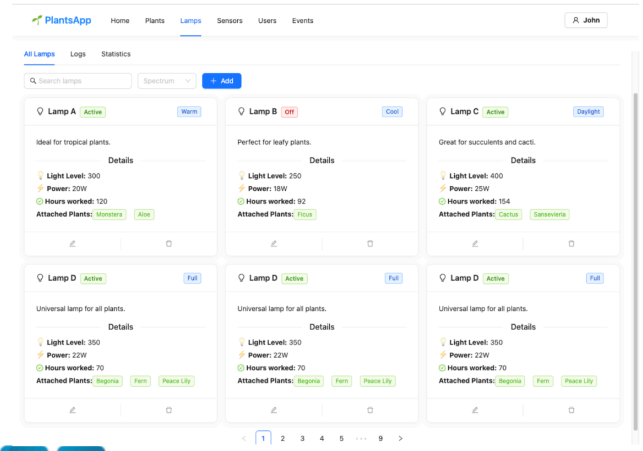
Основні елементи:

- Plant Growth Over Time — графік росту рослин за датами.
- Watered/Not Watered — діаграма поливу за останні дні.
- Plant Health Status — кругова діаграма стану рослин (здорові, потребують поливу, хворі).
- Recent Events — список останніх подій, пов'язаних із поливом, сенсорами та станом рослин.

Інтерфейс побудований на React та Ant Design.

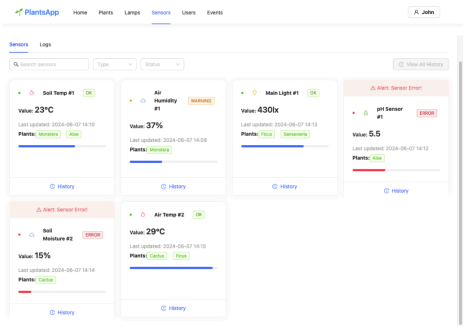


Інтерфейс користувача (управління лампами)



Цей фрагмент показує сторінку керування лампами: для кожної лампи відображаються параметри (потужність, світло, час роботи), статус, спектр і прикріплені рослини. Реалізовано пошук, фільтр, редагування, видалення та додавання нових ламп.

Інтерфейс користувача (дашборд із сенсорами)



На сторінці сенсорів відображаються всі активні пристрої моніторингу, що збирають дані про стан середовища — температуру, вологість, освітлення, pH грунту тощо. Для кожного сенсора показується актуальне значення, дата останнього оновлення, прив'язані рослини та статус роботи (OK, Warning або Error). Додатково доступний перегляд історії вимірювань.

16

Тестування (Ручне тестування API через Swagger)

№	Опис тесту	Роль користувача	Очікуваний результат	Результат
1	Авторизація через форму входу	Усі	Система створює сесію і перенаправляє на /home	✅
2	Доступ до сторінки /users без ролі admin	Звичайний	Отримано помилку доступу (403)	✅
3	Додавання нового сенсора	Технік	Сенсор з'являється в загальному списку	✅
4	Виведення списку рослин	Усі	Відображено перелік з актуальними даними	✅
5	Автоматичний редірект на login при втраті сесії	Усі	Користувача повертає на сторінку авторизації	✅
6	Створення дубліката користувача	Адмін	Виведено повідомлення про помилку	✅
7	Некоректна відповідь від сенсора (симуляція)	Технік	Відображено попередження/помилку	✅

17

Тестування (Unit-тестування NestJS)

№	Модуль	Тест	Очікуваний результат	Результат
1	AuthService	login() з валідними даними	Повертає сесію	☑
2	UserService	findById() з неіснуючим ID	Повертає null	☑
3	SensorService	createSensor() з валідними даними	Створення сенсора	☑
4	JSON	parse() з порожнім файлом	Повертає помилку	☑

18

Тестування (E2E-тестування - NestJS + Supertest)

№	Сценарій	Очікуваний результат	Результат
1	Користувач проходить реєстрацію та логін	Видається сесія, redirect на /home	✅
2	Авторизований запит на /me	Повертається об'єкт користувача	✅
3	Невалідний запит на /lamps без сесії	Отримано статус 401	✅
4	Повний сценарій: створення лампи – перегляд	Лампа створена та відображається	✅

Тестування (Тестування фронтенду ручне)

№	Сторінка	Тест	Очікуваний результат	Результат
1	/login	Успішний логін з валідними даними	Перенаправлення на /	✅
2	/plants	Завантаження списку рослин	Відображається таблиця	✅
3	/users	Відображення тільки для admin	Приховано у меню для інших ролей	✅
4	/sensors	Відображення попереджень для сенсорів з помилкою	Виводиться блок із "Sensor Error"	✅
5	/ (Dashboard)	Відображення графіків росту та поливу рослин	Побудова графіків	✅

Підсумки

У результаті виконаної роботи було створено повноцінну клієнтську частину системи для керування освітленням рослин, яка забезпечує авторизацію через сесії, роботу з API та зручне візуальне відображення даних у режимі реального часу.

Отримані результати є цілком реалістичними для подальшого практичного застосування в умовах тепличних господарств або дослідницьких лабораторій.

У майбутньому система може бути розширена новими типами пристроїв, мобільною версією та модулями аналітики для прогнозування стану рослин.

ДОДАТОК В
Специфікація ПЗ

1 ВСТУП

1.1 Огляд продукту

Клієнтська частина програмної системи призначена для візуалізації та управління процесами освітлення рослин у тепличних умовах. Вона забезпечує зручний інтерфейс для користувачів різних ролей (адміністратор, технічний персонал, звичайний користувач) та взаємодіє з серверною частиною через REST API. Застосунок дозволяє переглядати показники сенсорів у реальному часі, керувати рослинами, лампами, сенсорами, а також здійснювати операції з обліковими записами користувачів. Особливу увагу приділено безпечній авторизації через сесії з використанням httpOnly cookie.

1.2 Мета

Метою розробки є створення клієнтської частини вебзастосунку для моніторингу та керування освітленням рослин, яка дозволить користувачам зручно працювати з системою через браузер, здійснювати автентифікацію, переглядати дані сенсорів, отримувати попередження, та управляти пов’язаними об’єктами.

1.3 Межі

Функціональні межі клієнтської частини:

– Розроблено за допомогою React і TypeScript;

– Інтерфейс побудований з використанням компонентів Ant Design;

– Для маршрутизації застосовано React Router;

– Авторизація користувачів реалізована через сесії з httpOnly cookie (без токенів);

– Робота з API організована через RTK Query із централізованою обробкою помилок і кешуванням;

– Підтримка розмежування прав доступу на рівні інтерфейсу (protected routes, меню);

– Інтерфейс адаптовано під рольову модель доступу: різні можливості для адміністратора, техніка та звичайного користувача;

– Відображення стану системи, подій, сенсорів, рослин і ламп у режимі, наближеному до реального часу;

– Підтримка імпорту/експорту користувачів через JSON.

1.4 Посилання

У процесі розробки використовувались такі ресурси:

– https://react.dev

– https://redux-toolkit.js.org/rtk-query/overview

– https://ant.design/docs/react/introduce

– https://reactrouter.com/en/main/start/tutorial

– https://www.npmjs.com/package/express-session

– https://github.com/tj/connect-redis

1.5 Означення та абревіатури

Користувач — особа, що працює з вебінтерфейсом. Може мати одну з ролей: адміністратор, технік або звичайний користувач.

API — інтерфейс програмного взаємодії клієнтської частини з серверною.

Сесія — механізм збереження стану користувача між HTTP-запитами, реалізований через Redis і httpOnly cookies.

RTK Query — інструмент із Redux Toolkit для запитів до API та кешування.

Ant Design — компонентна бібліотека для створення сучасного інтерфейсу.

SPA (Single Page Application) — архітектурний підхід, за якого вся логіка інтерфейсу працює на одній сторінці.

JSON — формат файлів, що використовується для імпорту/експорту користувачів.

ProtectedRoute — компонент, який обмежує доступ до певних сторінок залежно від статусу автентифікації та ролі користувача.

2 ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

Основні перспективи продукту:

– Фронтенд-частина програмної системи має потенціал для подальшого розвитку в кількох напрямках:

– реалізація мобільного застосунку для Android/iOS для зручного доступу з будь-якої точки;

– додавання інтеграції з іншими типами пристроїв (зволожувачі, системи поливу тощо);

– вдосконалення системи візуалізації, зокрема аналітики стану рослин у вигляді динамічних графіків і діаграм;

– підтримка повідомлень у реальному часі через WebSocket або push-нотифікації;

– інтеграція з хмарними сервісами для централізованого зберігання і синхронізації стану між кількома пристроями.

2.2 Функції продукту

Інтерфейс користувача забезпечує такі можливості:

– авторизація через сесію (httpOnly cookie);

– перевірка ролей і контроль доступу до сторінок;

– перегляд, створення, редагування та видалення користувачів (доступно лише адміністраторам);

– керування рослинами: створення, перегляд, редагування, видалення;

– робота з сенсорами: додавання, редагування, прив'язка до рослин, перегляд, видалення;

– керування джерелами освітлення: додавання, налаштування параметрів, перегляд, видалення;

– перегляд даних сенсорів у реальному часі;

– візуалізація показників (температура, вологість, освітленість тощо);

– відображення попереджень щодо стану рослин;

– експорт і імпорт користувачів у форматі CSV.

2.3 Характеристики користувачів

2.3.1 Адміністратори

– мають повний доступ до управління системою;

– керують ролями користувачів та їх обліковими записами;

– працюють з інтерфейсом експорту/імпорту CSV;

– мають базову технічну підготовку.

2.3.2 Технічний персонал

– обслуговує сенсори, освітлення та рослини;

– використовує систему для додавання/редагування об’єктів;

– очікує простоти інтерфейсу без перевантаження зайвими функціями.

2.3.3 Кінцеві користувачі (фермери/аграрії)

– переглядають поточний стан рослин та сенсорні показники;

– отримують повідомлення про критичні умови;

– зацікавлені в інтуїтивному, мобільному та візуально зрозумілому інтерфейсі.

2.4 Загальні обмеження

На даному етапі присутні такі обмеження:

– клієнтська частина доступна лише через веб-браузер (мобільного застосунку наразі немає);

– доступ до сторінок залежить від ролі користувача;

– підтримуються лише українська та англійська локалізації;

– frontend працює лише за наявності доступу до API-сервера.

2.5 Припущення й залежності

На даному етапі припущення та залежності такі:

– клієнтський застосунок очікує, що сервер видає сесію через cookie при логіні;

– вся логіка авторизації та перевірки прав здійснюється на бекенді, клієнт лише реагує на статуси;

– передбачається наявність стабільного API з передбачуваними відповідями (JSON);

– RTK Query використовується для комунікації з API, тому API має підтримувати REST-ендпоінти з відповідними методами;

– дані сенсорів оновлюються через бекенд у реальному часі або з невеликою затримкою.

3 КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Клієнтська частина розроблена як односторінковий веб застосунок (SPA) на базі React. Інтерфейс забезпечує зручну навігацію для різних типів користувачів. Основні елементи UI:

– сторінка входу з авторизацією через сесію;

– сторінки для керування рослинами, сенсорами, освітленням, зчитуваннями;

– адміністративна панель для керування користувачами та ролями;

– реалізовано підтримку української та англійської мов;

– доступ до REST API здійснюється через RTK Query.

3.1.2 Апаратний інтерфейс

Вимоги до апаратного інтерфейсу:

– сучасний браузер з підтримкою JavaScript ES6;

– мінімум 2 ГБ оперативної пам’яті;

– підключення до мережі інтернет.

3.1.3 Програмний інтерфейс

Серверна частина реалізована на основі фреймворку NestJS із застосуванням TypeORM і PostgreSQL. Основні характеристики:

– реалізовано REST API для взаємодії з клієнтською частиною та IoT-пристроями;

– впроваджено сесійну авторизацію зберіганням сесій у Redis (через cookie, без використання токенів);

– для передачі сенсорних зчитувань у реальному часі використовується протокол MQTT.

Клієнтська частина створена з використанням React та TypeScript:

– для обміну даними з бекендом використовується RTK Query (Redux Toolkit Query);

– інтерфейс побудований з використанням Ant Design, що забезпечує сучасний адаптивний вигляд;

– реалізовано підтримку рольової моделі, багатомовності (українська/англійська);

– інтерфейс дозволяє користувачам керувати рослинами, сенсорами, освітлювальними пристроями та переглядати зчитування в реальному часі;

– сесія зберігається у браузері як HttpOnly cookie, що забезпечує додатковий захист.

Таким чином, клієнт та сервер взаємодіють через REST API та захищену сесію, забезпечуючи зручний та безпечний веб-інтерфейс для аграрних користувачів.

3.1.4 Комунікаційний протокол

У якості комунікаційних протоколів було використано:

– HTTPS для захищеного обміну даними між фронтендом і сервером;

– MQTT для комунікації з IoT-пристроями;

– Сесії реалізовані через express-session з RedisStore, cookie-підхід замість JWT.

3.1.5 Обмеження пам’яті

Клієнтська частина є вебзастосунком, який не потребує встановлення на пристрої користувача — достатньо лише сучасного браузера. Це знижує вимоги до апаратного забезпечення користувача і дозволяє працю

3.1.6 Операції

Основні функціональні операції системи:
– вхід/вихід користувача;
– керування користувачами та ролями;
– створення/редагування рослин, сенсорів та освітлення;
– зчитування показників з сенсорів;
– перегляд попереджень та журналу зчитувань;
– імпорт/експорт даних у форматі JSON.

3.1.7 Функції продукту

Програмне забезпечення має такі функції:
– авторизація та керування сесією;
– інтерфейс для керування даними;
– обробка сенсорних зчитувань у реальному часі;
– автоматичне регулювання освітлення;
– рольова модель доступу.

3.1.8 Припущення й залежності

Припущення:
– користувачі мають базові навички користування ПК;
– пристрої підключені до локальної мережі;

– дані з сенсорів є коректними;

Залежності:

– доступність серверної частини;

– стабільна робота MQTT брокера;

– коректність налаштувань Redis та бази даних;

– справність IoT-пристроїв.

3.2 Властивості програмного продукту

Розроблений програмний продукт дозволяє здійснювати моніторинг і автоматизоване керування освітленням для вирощування рослин у контрольованому середовищі на основі показників сенсорів. Система включає зручний веб-інтерфейс, що адаптований для різних типів користувачів (адміністратор, технік, звичайний користувач, підтримка).

Основні властивості продукту:

– клієнтська частина реалізована як SPA (Single Page Application) на основі React з використанням TypeScript і RTK Query для інтеграції з API;

– зчитування параметрів середовища (вологість, температура, освітлення) відображаються на фронтенді в реальному часі завдяки інтеграції з MQTT;

– реалізовано CRUD-операції для управління рослинами, сенсорами та освітлювальними пристроями через зручні інтерфейси;

– підтримується автоматичне керування освітленням відповідно до даних із сенсорів і встановлених параметрів рослин;

– система попереджень інформує користувача про критичні відхилення у стані середовища (наприклад, нестача світла чи перевищення вологості);

– реалізовано рольову модель доступу з гнучким керуванням правами користувачів через адміністративну панель;

– є можливість експорту та імпорту облікових записів користувачів у форматі CSV;

– автентифікація реалізована на основі сесій з використанням Redis, що забезпечує безпечний і масштабований контроль доступу;

– клієнтська частина має адаптивний дизайн і підтримку української та англійської мов для зручності користувачів.

3.3.1 Надійність

Ціль: система повинна стабільно працювати в режимі реального часу, обробляючи сенсорні дані та виконуючи відповідні дії без збоїв.

Метрика: час простою системи не повинен перевищувати 0.1% на місяць при типовому навантаженні. Передбачено обробку критичних ситуацій на фронтенді для інформування користувача про несправності.

3.3.2 Доступність

Ціль: забезпечити постійний доступ до системи через веб-інтерфейс та гарантувати відображення актуальних даних із сенсорів.

Метрики:

– доступність клієнтської та серверної частини — не менше 99.9% на місяць;

– доступ з будь-якого пристрою через браузер у локальній або корпоративній мережі;

– мінімізація часу завантаження інтерфейсу для зручної роботи користувачів.

3.3.3 Безпека

Ціль: гарантувати захист конфіденційних даних користувачів і цілісність операцій керування освітленням.

Метрики:

– використання хешування паролів на сервері;

– автентифікація та авторизація реалізовані через сесії (зберігаються у Redis);

– cookie з прапорцем HttpOnly;

– передача даних через HTTPS;

– обробка помилок доступу на фронтенді та повідомлення користувача.

3.3.4 Супроводжуваність

Ціль: система повинна легко підтримуватись і розширюватись без необхідності повної переробки.

Метрики:

– розділення логіки на фронтенді за допомогою модулів і Redux-слайсів;

– дотримання єдиних принципів оформлення коду (ESLint, Prettier);

– документація API через Swagger;

– покриття основних частин коду unit та e2e тестами.

3.3.5 Переносимість

Ціль: забезпечити можливість розгортання клієнтської та серверної частини на різних платформах (локальні ПК, сервери, хмари).

Метрики:

– використання кросплатформених технологій: React + NestJS + PostgreSQL;

– зручна збірка фронтенду через Vite/webpack;

– підтримка Docker для ізольованого розгортання.

3.3.6 Продуктивність

Ціль: швидка реакція системи на зміни в середовищі, забезпечення плавного інтерфейсу користувача.

Метрики:

– реакція на нові зчитування із затримкою не більше 1 секунди;

– сервер обробляє щонайменше 1000 запитів на хвилину;

– фронтенд показує дані без помітних затримок навіть на пристроях із середніми характеристиками.

3.4 Вимоги бази даних

У системі використовуються дві бази даних: основна реляційна база PostgreSQL та кешуюче сховище Redis.

PostgreSQL призначена для постійного зберігання структурованих даних. У ній зберігається така інформація:

– облікові записи користувачів та їхні ролі

– інформація про рослини, включаючи стадії росту та вимоги до освітлення

– дані сенсорів: тип, стан, прив’язка до рослин

– параметри освітлювальних пристроїв: яскравість, спектр, стан

– зчитування з сенсорів: температура, вологість, освітленість

– попередження щодо відхилення параметрів від встановлених норм

Redis використовується для тимчасового зберігання сесій користувачів. Це забезпечує:

– швидкий доступ до інформації про автентифікованих користувачів

– зменшення навантаження на основну базу даних

– підтримку TTL (часу життя сесії) та автоматичне очищення неактивних сесій

– надійну роботу механізму сесійної авторизації з використанням httpOnly cookie

3.5 Інші вимоги

Інші вимоги до системи:

– підтримка багатомовного інтерфейсу (українська та англійська мови);

– зберігання користувацьких сесій у Redis для підвищення швидкодії;

– фронтенд адаптований для роботи на пристроях із різною роздільною здатністю;

– інтеграція з MQTT для обміну даними з фізичними пристроями в реальному часі;

– забезпечення доступу до системи тільки авторизованим користувачам відповідно до їхніх ролей.