МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет Навчально-науковий інститут економіки та бізнес-освіти

Кафедра Економіки та цифрового бізнесу

Спеціальність 122 «Комп'ютерні науки»

Форма навчання Денна

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Александрова Тихона Анатолійовича
(прізвище, ім'я, по батькові здобувача)

на тему Розробка веб-додатка для моніторингу та диспетчеризації термостатної системи промислового підприємства
(повна назва теми)

за матеріалами _____
(повна назва бази дослідження)

науковий керівник к.п.н., доцент _____ Павлиш Т.Г.
(наук. ступінь, вчене звання) *(підпис)* *(прізвище, ініціали)*

Робота допущена до захисту в ЕК
Протокол засідання кафедри
від червня 2026 р. №
Завідувач кафедри _____
(підпис)
к.е.н., доцент Радько В.М.
наук. ступень, вчене звання прізвище, ініціали

Кривий Ріг – 2026

ЗАТВЕРДЖЕНО
Наказ Міністерства освіти і науки, молоді та спорту України
29 березня 2012 року № 384
Форма № Н-9.01

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ ТА БІЗНЕС-ОСВІТИ
(повне найменування вищого навчального закладу)

Кафедра економіки та цифрового бізнесу
Освітній ступінь бакалавр
Спеціальність 122 Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри _____В.М. Радько

"30" березня 2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ ЗДОБУВАЧУ

_____Александрову Тихону Анатолійовичу_____

1. Тема роботи Розробка веб-додатка для моніторингу та диспетчеризації термостатної системи промислового підприємства
науковий керівник роботи ____к.п.н., доцент, Павлиш Т.Г.
затверджені наказом вищого навчального закладу від «23» березня 2026 р. № 193-ст (д/ф)

2. Строк подання здобувачем роботи 29.05.2026р.

3. Зміст кваліфікаційної бакалаврської роботи, об'єкт, предмет та мета дослідження:

Розділ 1 - Теоретичні основи та аналіз програмних засобів моніторингу термостатних систем промислових підприємств
Розділ 2 - Інформаційне та математичне забезпечення веб-додатка для моніторингу та диспетчеризації термостатної системи промислового підприємства
Розділ 3 - Розробка програмного забезпечення веб-додатка моніторингу та диспетчеризації процесу термомеханічного зміцнення арматури

Об'єкт дослідження – процес моніторингу та диспетчеризації системи термомеханічного зміцнення арматури на промисловому підприємстві

Предмет дослідження – методи, моделі та програмні засоби розробки веб-орієнтованих систем моніторингу й диспетчеризації температурних режимів промислових технологічних процесів

Мета кваліфікаційної бакалаврської роботи – розробка веб-додатка для моніторингу та диспетчеризації системи термомеханічного зміцнення арматури з використанням сучасних веб-технологій та засобів передачі даних у реальному часі
4. Дата видачі завдання 03.04.2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів роботи	Відмітка керівника про виконання етапів (дата, підпис)
1	Підготовка розділу 1	до 17.04.2026р.	
2	Підготовка розділу 2	до 08.05.2026р.	
3	Підготовка розділу 3	до 25.05.2026р.	
4	Реєстрація завершеної кваліфікаційної роботи	до 29.05.2026р.	
5	Отримання відгуку від наукового керівника	04.06.2026р.	
6	Отримання зовнішньої рецензії	05.06.2026р.	
7	Попередній захист кваліфікаційної роботи на кафедрі	08.06.2026р.	
8	Перевірка кваліфікаційної роботи на плагіат	12.06.2026р. д/ф 18.06.2026р. з/ф	
9	Допуск кафедрою кваліфікаційної роботи до захисту	12.06.2026р. д/ф 18.06.2026р. з/ф	
10	Підготовка студента до захисту в ЕК	до 19.06.2026р. д/ф до 23.06.2026р з/ф	

Завдання підготував науковий керівник _____ Павлиш Т.Г.
(підпис) (прізвище та ініціали)

Завдання одержав здобувач  Александров Т.А.
(підпис) (прізвище та ініціали)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ACID – Atomicity, Consistency, Isolation, Durability (атомарність, узгодженість, ізольованість, стійкість)

API – Application Programming Interface (інтерфейс програмування застосунків)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

CRUD – Create, Read, Update, Delete (створення, читання, оновлення, видалення)

ER-діаграма – Entity-Relationship diagram (діаграма «сутність-зв’язок»)

HTML – HyperText Markup Language (мова розмітки гіпертексту)

HTTPS – HyperText Transfer Protocol Secure (захищений протокол передачі гіпертексту)

JSON – JavaScript Object Notation (формат обміну даними JavaScript)

JWT – JSON Web Token (веб-токен JSON)

Node.js – Node JavaScript (середовище виконання JavaScript)

PDF – Portable Document Format (портативний формат документів)

QR-код – Quick Response code (швидкодійний код)

REST API – Representational State Transfer Application Programming Interface (інтерфейс програмування застосунків у стилі REST)

SPA – Single Page Application (односторінковий застосунок)

SQL – Structured Query Language (структурована мова запитів)

SQLite – Structured Query Language Lite (легка версія SQL)

UI – User Interface (інтерфейс користувача)

UML – Unified Modeling Language (уніфікована мова моделювання)

UX – User Experience (досвід користувача)

VIP – Very Important Person (дуже важлива особа)

WebSocket – протокол повнодуплексного зв’язку в реальному часі

РЕФЕРАТ

Кваліфікаційна робота складається із: 64 с., 11 рис., 2 табл., 25 джерел.

Тема роботи: «Розробка веб-додатка для моніторингу та диспетчеризації термостатної системи промислового підприємства».

Об’єкт дослідження – процес моніторингу та диспетчеризації системи термомеханічного зміцнення арматури на промисловому підприємстві.

Предмет дослідження – методи, моделі та програмні засоби розробки веб-орієнтованих систем моніторингу й диспетчеризації температурних режимів промислових технологічних процесів.

Мета роботи – розробка веб-додатка для моніторингу та диспетчеризації системи термомеханічного зміцнення арматури з використанням сучасних веб-технологій та засобів передачі даних у реальному часі.

У роботі проведено аналіз предметної області та особливостей функціонування системи термомеханічного зміцнення арматури, досліджено існуючі системи моніторингу й диспетчеризації промислових процесів, визначено їх переваги та недоліки. На основі проведеного аналізу сформульовано функціональні та нефункціональні вимоги до програмного забезпечення.

Розроблено інформаційну модель системи, спроєктовано структуру бази даних та математичну модель генерації телеметричних даних для імітації роботи датчиків температури. Обґрунтовано вибір клієнт-серверної архітектури та технологічного стеку, що включає FastAPI, WebSocket, SQLite та сучасні веб-технології для створення користувацького інтерфейсу.

У результаті виконання роботи реалізовано веб-додаток, який забезпечує збір, передачу, зберігання та візуалізацію телеметричних даних у режимі реального часу. Система підтримує відображення графіків температурних параметрів, автоматичне формування повідомлень про відхилення від заданих режимів, ведення історії показників та розмежування прав доступу користувачів. Практичне значення роботи полягає у створенні працездатного програмного рішення, яке може бути використане для підвищення ефективності контролю

технологічних процесів, оперативності прийняття рішень та зниження ризику виникнення виробничих відхилень на промислових підприємствах.

БАЗА ДАНИХ, ВЕБ-ДОДАТОК, ДИСПЕТЧЕРИЗАЦІЯ, ІНФОРМАЦІЙНА СИСТЕМА, МОНІТОРИНГ, ПРОМИСЛОВЕ ПІДПРИЄМСТВО, ТЕЛЕМЕТРИЧНІ ДАНІ, ТЕРМОМЕХАНІЧНЕ ЗМІЦНЕННЯ АРМАТУРИ, FASTAPI, WEBSOCKET.

ВСТУП

Актуальність теми кваліфікаційної роботи зумовлена активним розвитком цифрової трансформації металургійного виробництва в рамках концепції Industry 4.0. Сучасні підприємства стикаються з необхідністю підвищувати якість продукції, знижувати енергоспоживання, забезпечувати стабільність технологічних процесів і швидко реагувати на будь-які відхилення параметрів [1, 3]. Одним із найбільш відповідальних і енергоємних процесів у виробництві будівельної арматури є термомеханічне зміцнення (ТМЗ) арматури – комплексна термічна обробка, під час якої сталь проходить високотемпературний нагрів, витримку та прискорене охолодження за строго регламентованими режимами. Якість виконання цього процесу безпосередньо визначає механічні властивості готової арматури та її відповідність вимогам ДСТУ 3760:2018 і ГОСТ 10884-94 [3].

Питанням дослідження технології термомеханічного зміцнення арматури та автоматизації температурних режимів присвячені роботи багатьох українських і зарубіжних науковців. Фундаментальні аспекти фазових перетворень при ТМЗ висвітлені в працях О. В. Вакуленко та колективу НМетАУ. Проблеми моніторингу та диспетчеризації промислових процесів на основі IoT і SCADA-систем розглянуто в роботах А. І. Глушакова [1], Є. М. Гунька [2], В. І. Крижанівського [4] та інших дослідників. Разом з тим, аналіз літератури показує, що недостатньо уваги приділяється саме спеціалізованим веб-орієнтованим системам диспетчеризації багатозонних ліній ТМЗ з урахуванням реальних умов металургійного виробництва (запиленість, електромагнітні перешкоди, висока динаміка процесу) [1, 2, 4].

Система термомеханічного зміцнення арматури є складним багатозонним термостатним комплексом, який дуже чутливий до найменших відхилень температурних режимів. Навіть короткочасне порушення заданих параметрів призводить до неоднорідної структури металу, зниження міцності, пластичності

та виходу продукції за межі нормативних вимог. У зв’язку з цим особливої актуальності набуває створення сучасної системи моніторингу та диспетчеризації, яка забезпечує збір даних у реальному часі, візуалізацію процесів, оперативне сповіщення про відхилення та централізоване управління всіма технологічними зонами [1, 4].

Традиційні локальні SCADA-системи вже не задовольняють сучасним вимогам щодо віддаленого доступу, централізованої диспетчеризації, аналізу великих обсягів телеметричних даних та інтеграції з MES/ERP-системами підприємства [4, 5]. Усе це обумовлює необхідність розробки спеціалізованого веб-додатка, який поєднує надійність промислових протоколів, зручний веб-інтерфейс і потужні аналітичні можливості.

Об'єкт дослідження – процес моніторингу та диспетчеризації системи термомеханічного зміцнення арматури на промисловому підприємстві.

Предмет дослідження – методи, моделі та програмні засоби розробки веб-додатка моніторингу і диспетчеризації температурних режимів системи термомеханічного зміцнення арматури.

Мета кваліфікаційної роботи полягає в розробці веб-додатка для моніторингу та диспетчеризації системи термомеханічного зміцнення арматури промислового підприємства на основі сучасних веб- та IoT-технологій.

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

– проаналізувати предметну область та особливості функціонування системи термомеханічного зміцнення арматури;

– здійснити огляд наявних аналогів та систем моніторингу температурних режимів;

– сформулювати постановку задачі та обґрунтувати вимоги до веб-додатка диспетчеризації;

– виконати концептуальне проєктування системи та побудувати інформаційну модель;

– здійснити математичне моделювання алгоритмів збору та обробки телеметричних даних;

– спроєктувати архітектуру бази даних;

– реалізувати інтерактивний інтерфейс та основний функціонал веб-додатка;

– провести комплексне тестування та розробити рекомендації щодо впровадження.

У роботі були використані методи системного аналізу, порівняльного аналізу, математичного моделювання, об’єктно-орієнтованого проєктування (UML) та методи програмної інженерії.

Наукова новизна одержаних результатів полягає в розробці спеціалізованої архітектури веб-додатка, адаптованої до особливостей багатозонної системи термомеханічного зміцнення арматури з урахуванням вимог реального металургійного виробництва, кібербезпеки та економічної ефективності.

Практична значущість роботи полягає в можливості реального впровадження розробленого веб-додатка на промислових підприємствах для підвищення оперативності диспетчеризації, зниження браку продукції, зменшення енергоспоживання та оптимізації всього технологічного процесу.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. У першому розділі проаналізовано стан проблемної області. Другий розділ присвячено інформаційному та математичному забезпеченню. Третій розділ містить опис програмного та технічного забезпечення розробленого рішення.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ ПРОГРАМНИХ ЗАСОБІВ МОНІТОРИНГУ ТЕРМОСТАТИЧНИХ СИСТЕМ ПРОМИСЛОВИХ ПІДПРИЄМСТВ

1.1. Аналіз предметної області та особливостей функціонування системи термомеханічного зміцнення арматури

Термомеханічне зміцнення арматури реалізовано як багатостадійний процес, що поєднує гарячу пластичну деформацію та контрольоване прискорене охолодження з подальшим самовідпуском. Було проаналізовано, що на початковій стадії квадратні заготовки (150×150 мм) нагріваються в печах перезарядки до 1150–1250 °C для отримання аустенітної структури з розміром зерна ASTM 7–8 [7]. Після чернової та чистової груп клітей з формуванням ребер за допомогою трайб-апаратів пруток надходить до ТМТ-касети (quenching box), оснащеної 3–5 секціями форсунок високого тиску типу V-jet (тиск 12–25 бар, витрата 150–350 м³/год).

У першій секції касети здійснюється інтенсивне поверхневе охолодження зі швидкістю 80–150 °C/с до температури поверхні 450–620 °C, що ініціює мартенситне перетворення поверхневого шару товщиною 1,5–3,5 мм. Осердя прутка зберігає температуру 780–920 °C, забезпечуючи дифузію тепла назовні та самовідпуск мартенситу з утворенням відпущеного мартенситу та бейніту [7]. Фінальна стадія – повітряне охолодження на холодильному столі зі швидкістю 1–3 °C/с. Ключовими технологічними параметрами є температура входу в касету (Твх = 950–1050 °C), швидкість прокатки (10–18 м/с), тиск і витрата води, геометрія форсунок.

Було досліджено, що стабільність механічних властивостей ($\sigma 0{,}2 \geq 500$ МПа, $\delta 5 \geq 16$ %) безпосередньо залежить від синхронізації зазначених параметрів. У джерелах [1; 4] підкреслюється, що традиційні SCADA-системи з

частотою опитування 0,5–1 Гц не забезпечують необхідної оперативності, що призводить до запізнення реакції диспетчера. Відхилення температури поверхні лише на 8–12 °C або тривалості перебування в зоні quenching на 2–4 секунди викликає неповне мартенситне перетворення, утворення мікротріщин, зниження умовної межі текучості на 12–18 % та перехід класу міцності на нижчий рівень (з A500S до A400S) [7].

Збір телеметрії (температура входу/виходу кожної секції, ΔT, швидкість, тиск, витрата) щосекунди з частотою не менше 1 Гц визнано критичним, оскільки навіть короткочасні коливання через гідроакумулятори, забруднення форсунок або варіації хімічного складу (C 0,18–0,25 %, Mn 0,6–1,2 %) призводять до неоднорідності структури по довжині прутка та масового браку [1; 4]. Аналіз підтверджує, що існуючі промислові мережі не інтегровані з сучасними веб-інтерфейсами реального часу, що обґрунтовує необхідність розробки спеціалізованої системи диспетчеризації.

У ході аналізу встановлено, що апаратне забезпечення моніторингу температурних режимів на прокатному стані включає комплекс високотехнологічних датчиків, адаптованих до екстремальних цехових умов. Зокрема, для вимірювання температури поверхні прутка застосовуються двоколірні оптичні пірометри (ratio pyrometers), принцип дії яких ґрунтується на співвідношенні інтенсивності випромінювання в двох спектральних діапазонах, що дозволяє компенсувати вплив окалини, пари та пилу. Електромагнітні витратоміри води (типу магнітно-індуктивних) забезпечують точне визначення витрати охолоджувальної рідини в кожній секції ТМТ-касети незалежно від її провідності. Перетворювачі тиску (pressure transducers з мембранними сенсорами) реєструють значення в діапазоні 0–40 бар з частотою до 10 Гц. Було виявлено, що в умовах інтенсивної вібрації (до 50 Гц), запиленості та присутності водяної пари зазначені пристрої оснащуються системами примусового продування стисненим повітрям, захисними кожухами з кварцовим склом та

системами примусового охолодження, що гарантує стабільність сигналу відповідно до вимог промислових мереж Industry 4.0 [1; 4].

Розглянуто також особливості функціонування мікроструктури металу при порушенні режимів. Встановлено, що при нерівномірній подачі води або падінні тиску в форсунках відбувається порушення градієнта охолодження, внаслідок чого поверхневий шар не досягає критичної швидкості охолодження 80–150 °C/с. У результаті замість формування відпущеного мартенситу з високою щільністю дислокацій утворюється ферито-перлітна суміш з грубозернистою структурою, що супроводжується зниженням мікротвердості поверхні з 420–480 HV до 280–340 HV. На мікрорівні спостерігається неповне розчинення карбідів, збільшення об’єму залишкового аустеніту та утворення мікротріщин вздовж меж зерен колишнього аустеніту, що суттєво погіршує ударну в’язкість та пластичність [7]. Аналогічні дефекти фіксуються при коливаннях швидкості прокатки, коли час перебування в зоні quenching відхиляється від оптимального на 1,5–3 секунди.

Особлива увага приділена ролі диспетчера в оперативному управлінні. Було встановлено, що затримка відображення телеметричних даних у застарілих SCADA-інтерфейсах на 8–15 секунд призводить до запізнілого коригування параметрів (зниження/підвищення тиску, зміна швидкості клітей), що унеможливлює запобігання браку в реальному масштабі часу. Унаслідок цього ціла партія арматури (20–50 тонн) може бути переведена з класу міцності A500S до A400S через невідповідність механічних характеристик вимогам ДСТУ, що спричиняє перемаркування, зниження ціни реалізації на 15–25 % та прямих економічних втрат підприємства в розмірі кількох мільйонів гривень на одну зміну. Аналіз, наведений у збірнику, підтверджує, що відсутність інтегрованих веб-дашбордів реального часу значно підвищує ризики технологічних порушень та загальну собівартість продукції [3].

Таким чином, комплексний аналіз предметної області та функціональних особливостей системи термомеханічного зміцнення арматури свідчить про нагальну потребу у створенні сучасного веб-додатка диспетчеризації, здатного

інтегрувати дані від зазначених датчиків, моделювати мікроструктурні процеси та забезпечувати миттєву візуалізацію для оперативного прийняття рішень.

Було детально проаналізовано технологічний процес термомеханічного зміцнення арматури на прокатному стані. Після проходження чистової групи клітей пруток із температурою 950–1050 °C потрапляє в ТМТ-касету. Головна особливість цього обладнання – різке охолодження поверхні прутка водою під високим тиском. Саме завдяки цьому різкому перепаду температур формується особлива двофазна структура металу.

У першій секції касети вода подається через форсунки під тиском 12–25 бар. Швидкість охолодження поверхневого шару сягає 80–150 °C за секунду. У цей момент на поверхні відбувається мартенситне перетворення: аустеніт перетворюється на твердий мартенсит. Поверхневий шар товщиною 1,5–3,5 мм стає дуже міцним і твердим. Водночас серцевина прутка залишається гарячою (близько 780–920 °C) і зберігає пластичність. Далі під час руху по холодильному столу тепло з гарячої серцевини поступово виходить на поверхню і відбувається самовідпуск мартенситу. Завдяки цьому арматура отримує високу міцність на поверхні і достатню гнучкість усередині.

Якщо води подати замало або тиск буде недостатнім, поверхня не встигає охолонути до потрібної температури. Мартенсит утворюється неповністю, замість нього з’являється бейніт або троостит. Арматура виходить м’якою, не витримує потрібного класу міцності і йде в брак. Якщо ж води подати забагато або охолодження відбувається нерівномірно, поверхня переохолоджується. Виникають мікротріщини, метал стає крихким, а при випробуваннях на вигин 180° пруток ламається в зоні ребер. На практиці навіть кілька секунд неправильного режиму призводять до відбракування цілої партії в 20–40 тонн.

Саме тому диспетчеру потрібно бачити реальні показники температури, тиску води та швидкості прокатки практично миттєво. Затримка навіть на 3–5 секунд може призвести до того, що оператор не встигне вчасно відреагувати на відхилення. Старим SCADA-системам часто бракує швидкості оновлення та

зручності. Саме тому виникла необхідність у створенні сучасного веб-додатка, який передаватиме дані з мінімальною затримкою через WebSocket, відображатиме чіткі графіки і одразу показуватиме аварійні ситуації. Такий додаток дозволить диспетчеру оперативно коригувати тиск води або швидкість прокатки і не допускати браку.

Було проаналізовано питання інформаційної безпеки при впровадженні веб-додатка диспетчеризації в умовах металургійного виробництва. Промислова мережа цеху – це не просто офісна локальна мережа. Тут працює критичне технологічне обладнання, і будь-яке порушення може призвести до серйозних наслідків.

Перша серйозна загроза – несанкціонований доступ до панелі диспетчера. Якщо стороння особа отримає логін і пароль, вона зможе зайти в систему і змінити налаштування порогових значень температури або тиску. У реальному виробництві це може призвести до неправильного охолодження арматури і масового браку. Тому в системі обов’язково застосовується надійне криптографічне хешування паролів за допомогою bcrypt. Звичайне зберігання паролів у відкритому вигляді або слабке хешування неприпустиме, адже зламати такий захист набагато легше.

Друга небезпека – перехоплення або підміна телеметричних даних у мережі цеху (Man-in-the-Middle атаки). Якщо зловмисник зможе перехопити дані, що йдуть від датчиків до сервера, і підмінити значення температури, диспетчер побачить на екрані неправильну картину. Він може подумати, що все в нормі, хоча насправді процес уже йде з відхиленням. Щоб захиститися від таких атак, усі дані, включаючи WebSocket-з’єднання, повинні передаватися тільки через захищені протоколи – WSS (WebSocket Secure) та HTTPS. Це забезпечує шифрування трафіку і перевірку цілісності даних.

Третя загроза – DDoS-атаки на сервер або мережу цеху. Навіть короткочасне «зависання» системи диспетчеризації може призвести до втрати контролю над процесом. У металургії зупинка лінії навіть на 10–15 хвилин – це

великі фінансові збитки. Тому серверна частина, побудована на сучасному фреймворку FastAPI, має важливу перевагу. FastAPI працює асинхронно, добре масштабується і здатний одночасно обробляти велику кількість з'єднань без «зависань». На відміну від старих локальних програм і SCADA-систем, які часто «лягають» при великому навантаженні, FastAPI стабільно витримує потік даних від десятків і навіть сотень датчиків.

Таким чином, при розробці веб-додатка питання кібербезпеки не можна відкладати на потім. Використання сучасних механізмів захисту (bcrypt для паролів, WSS для передачі даних, JWT для сесій) дозволяє значно знизити ризики і зробити систему придатною для реального промислового використання.

1.2. Огляд наявних аналогів та систем моніторингу температурних режимів

Було проведено системний огляд існуючих рішень, які застосовуються для моніторингу температурних режимів у процесах термомеханічного зміцнення арматури на металургійних підприємствах. Установлено, що на більшості комбінатів України та країн СНД досі переважають класичні промислові SCADA-системи, зокрема Siemens WinCC, Schneider Electric EcoStruxure та Honeywell. Зазначені системи забезпечують надійний збір даних безпосередньо від польових датчиків і PLC-контролерів через протоколи OPC UA/DA, швидке виведення трендів і аварійних повідомлень на операторські станції, а також дозволяють оперативне коригування параметрів [4; 5]. Характерними перевагами цих рішень є висока надійність роботи в промислових умовах, глибока сумісність з наявним обладнанням прокатних станів та багаторічна апробація на об'єктах аналогічного типу.

Водночас під час аналізу виявлено суттєві обмеження класичних SCADA-систем. Було встановлено, що вони характеризуються слабкими можливостями віддаленого доступу, складністю інтеграції з MES- та ERP-системами

підприємства, обмеженими інструментами аналізу історичних даних та високою вартістю ліцензійного супроводження [4; 5]. Незважаючи на очевидні переваги в локальному середовищі, зазначені системи не забезпечують гнучкості, необхідної для сучасних вимог Industry 4.0.

З поширенням концепції Industrial Internet of Things (IIoT) набули поширення веб-орієнтовані IoT-платформи, серед яких ThingsBoard, Ignition SCADA, а також комбінації Grafana з часовими базами даних InfluxDB та TimescaleDB. У джерелах зазначається, що ці платформи підтримують збір даних через промислові протоколи MQTT та OPC UA, дозволяють створювати інтерактивні дашборди, налаштовувати автоматичні сповіщення та забезпечувати доступ до інформації з будь-якого пристрою [2; 1]. Однак характерною особливістю таких рішень є їх універсальність: вони недостатньо враховують специфіку багатозонної системи термомеханічного зміцнення, високу динаміку процесу, потребу в жорсткій синхронізації десятків температурних зон та роботу в умовах сильної запиленості, високої температури та електромагнітних перешкод металургійного цеху [1; 2].

На підприємствах України переважають або власні розробки, або системи на базі програмованих логічних контролерів Siemens та Allen-Bradley. Було досліджено, що такі рішення демонструють високу надійність та сумісність з існуючим обладнанням, проте практично не мають сучасного веб-інтерфейсу, мобільного доступу та розширених аналітичних інструментів [5]. Комерційні рішення від ABB, Emerson та Endress+Hauser, хоча й пропонують розвинені функціональні можливості, відрізняються надвисокою вартістю впровадження, що робить їх недоступними для більшості середніх і великих металургійних підприємств України.

Для об’єктивної оцінки розглянутих аналогів було складено порівняльну таблицю 1.2.

Таблиця 1.1

Порівняльна характеристика систем моніторингу температурних режимів

Система	Тип архітектури	Швидкість оновлення даних (Latency)	Можливість кастомізації під ТМЗ	Вартість впровадження
Класичні SCADA (Siemens WinCC) [4; 5]	Локальна (Desktop/клієнт-сервер)	1–5 с	Низька (обмежена скриптами)	Висока (від 500 тис. грн + інфраструктура)
Веб-орієнтовані IoT-платформи (ThingsBoard / Grafana) [2; 1]	Веб / хмарна	0,5–3 с (з урахуванням cloud)	Середня (шаблони + плагіни)	Середня (підписка 8–25 тис. грн/міс.)
Наш розроблюваний веб-додаток	Повноцінна веб (FastAPI + React)	< 0,3 с (WebSocket, локальний/гібридний)	Висока (повна адаптація під багатозонний ТМЗ)	Низька (розробка в рамках дипломної + open-source компоненти)

Джерело: складено автором на основі аналізу [1; 2; 4; 5].

Аналіз таблиці 1.2 дозволяє зробити висновок, що класичні SCADA-системи залишаються прив’язаними до локальної інфраструктури та операторських станцій, що суттєво ускладнює масштабування та забезпечення віддаленого доступу. Веб-орієнтовані IoT-платформи, попри сучасний інтерфейс, демонструють додаткові затримки передачі даних та недостатній рівень кастомізації під специфіку процесу термомеханічного зміцнення. Розроблюваний веб-додаток, навпаки, поєднує переваги обох підходів: забезпечує мінімальну затримку, повну адаптацію під технологічні особливості ТМЗ-касети та суттєво нижчу вартість впровадження.

Таким чином, було встановлено, що на сучасному етапі практично відсутнє готове спеціалізоване програмне рішення, яке б одночасно поєднувало надійність промислових протоколів, сучасний зручний веб-інтерфейс, можливість віддаленого доступу та максимальну адаптацію саме під багатозонний процес термомеханічного зміцнення арматури [1; 2; 4; 5]. Існуючі системи або надто універсальні, або надто дорогі, або не забезпечують необхідного рівня

функціональності для ефективного управління термостатним комплексом. Виявлена прогалина обґрунтовує необхідність створення власного кастомного веб-додатка диспетчеризації.

Для обґрунтування вибору технології передачі даних у веб-додатку диспетчеризації було проведено детальний порівняльний аналіз основних мережевих протоколів, які можуть бути використані для моніторингу температурних режимів у реальному часі. У промислових умовах металургійного цеху, де дані від датчиків надходять з частотою 1–2 Гц і вище, вибір протоколу має критичне значення для швидкості оновлення інформації, навантаження на мережу та зручності роботи диспетчера.

Найпоширенішим і найпростішим у реалізації є підхід на базі HTTP REST API з постійним опитуванням сервера (polling). У цьому випадку клієнт (веб-браузер диспетчера) через певні проміжки часу (наприклад, кожні 500–1000 мс) надсилає запит на сервер, отримує актуальні дані і відображає їх. Такий метод добре працює для нечастих оновлень, але в умовах високочастотної телеметрії має суттєві недоліки. Кожен HTTP-запит містить велику кількість службових заголовків (headers), таких як User-Agent, Accept, Cookie, Authorization тощо. При частоті опитування 1 Гц один клієнт генерує близько 3600 запитів на годину. Якщо в системі працює кілька диспетчерських станцій, навантаження на мережу та сервер зростає в рази. Крім того, polling не є ефективним: навіть якщо на сервері немає нових даних, клієнт все одно отримує повну відповідь з HTTP-заголовками. Це створює зайвий трафік і збільшує затримку (latency) через необхідність встановлювати нове TCP-з’єднання для кожного запиту. У цехових умовах, де пропускна здатність мережі обмежена, такий підхід швидко стає неефективним.

Протокол MQTT (Message Queuing Telemetry Transport) – це легковаговий протокол, спеціально розроблений для IoT-систем. Він працює за моделлю «видавець–брокер–підписник» (publish-subscribe). Датчики публікують дані на брокер (наприклад, Mosquitto або HiveMQ), а клієнти підписуються на потрібні

топіки. Основними перевагами MQTT є низьке споживання трафіку, підтримка QoS (Quality of Service) рівнів і можливість роботи в умовах нестабільного зв’язку. Однак для веб-додатка диспетчеризації MQTT має важливе обмеження. Браузери не підтримують MQTT нативно. Для підключення браузера до MQTT-брокера необхідно використовувати додаткові технології, такі як MQTT over WebSocket або спеціальні бібліотеки (наприклад, MQTT.js). Це вимагає розгортання додаткового брокера, налаштування мостів і збільшує складність системи. У промисловому середовищі, де потрібна максимальна простота і надійність, наявність додаткової ланки (брокера) може стати потенційною точкою відмови.

WebSocket є оптимальним рішенням для задач, де потрібна передача даних у реальному часі. На відміну від HTTP, WebSocket встановлює постійне двостороннє (full-duplex) з’єднання між клієнтом і сервером після початкового рукостискання (handshake). Після встановлення з’єднання дані передаються у вигляді невеликих фреймів без повторної відправки HTTP-заголовків. Це значно зменшує навантаження на мережу. Один WebSocket-з’єднання може передавати дані з частотою 1–10 Гц практично без додаткових накладних витрат. Затримка передачі (latency) при цьому становить усього 50–300 мс залежно від якості мережі. WebSocket підтримується всіма сучасними браузерами нативно, що дозволяє створювати інтерактивні дашборди без використання додаткових брокерів. У нашому випадку протокол WebSocket дозволяє серверу (FastAPI) миттєво надсилати нові дані від імітатора датчиків до браузера диспетчера, забезпечуючи реальне оновлення графіка кожні 0,7 секунди. Крім того, WebSocket підтримує автоматичне відновлення з’єднання при тимчасовій втраті зв’язку, що критично важливо для цехових умов.

Таблиця 1.2

Порівняльна характеристика протоколів передачі даних для веб-додатка моніторингу

Протокол	Тип з'єднання	Навантаження на мережу (Overhead)	Затримка передачі (Latency)	Зручність інтеграції у веб-браузер
HTTP REST API (Polling)	Короткочасне (запит-відповідь)	Високе (повні HTTP-заголовки при кожному запиті)	Середня–висока (500–2000 мс)	Висока (нативна підтримка)
MQTT	Постійне (через брокер)	Низьке	Низька (50–300 мс)	Низька (потрібен MQTT.js + брокер)
WebSocket	Постійне двостороннє (full-duplex)	Дуже низьке	Дуже низька (< 300 мс)	Висока (нативна підтримка)

Аналіз таблиці показує, що протокол WebSocket пропонує найкраще поєднання низького навантаження на мережу, мінімальної затримки та зручності інтеграції безпосередньо в браузер. Саме тому для розроблюваного веб-додатка диспетчеризації було обрано WebSocket як основний протокол передачі телеметричних даних у реальному часі.

1.3. Постановка задачі та обґрунтування вимог до розроблюваного веб-додатка диспетчеризації

На основі проведеного аналізу предметної області, особливостей функціонування системи термомеханічного зміцнення арматури та огляду наявних аналогів було сформульовано основну задачу кваліфікаційної роботи. Головною задачею є розробка архітектури та програмної реалізації клієнт-серверного веб-додатка диспетчеризації, призначеного для моніторингу температурних режимів у реальному часі з можливістю імітації телеметричних

даних, інтерактивної візуалізації та оперативного реагування на відхилення технологічних параметрів.

Виходячи з виявлених недоліків існуючих систем, до розроблюваного веб-додатка висуваються такі функціональні вимоги:

– забезпечення безперервного збору та імітації телеметричних даних від віртуальних датчиків (температура входу/виходу з кожної секції ТМТ-касети, тиск і витрата води, швидкість прокатки);

– реалізація динамічної візуалізації температурних профілів у вигляді реал-тайм графіків, теплових карт і трендів з можливістю масштабування та вибору часових інтервалів;

– створення системи тригерів та алертів з візуальним та звуковим сповіщенням диспетчера при виході параметрів за встановлені технологічні межі (червоні/жовті статуси, push-повідомлення);

– автоматичне збереження всієї історії показників у реляційній базі даних для подальшого ретроспективного аналізу, формування звітів та оптимізації режимів.

Нефункціональні вимоги до системи визначено з урахуванням особливостей промислової експлуатації:

– продуктивність – затримка оновлення даних не більше 0,3 с за рахунок використання WebSocket-протоколу;

– надійність – забезпечення безперервної роботи навіть при тимчасовій втраті зв’язку з імітаційними датчиками та автоматичне відновлення сесії;

– масштабованість – можливість підключення додаткових ліній прокатки та користувачів без суттєвого зниження продуктивності;

– кросплатформеність і доступність – повноцінна робота в будь-якому сучасному браузері (Chrome, Edge, Firefox) без встановлення додаткового програмного забезпечення.

Окремо сформульовано вимоги до інтерфейсу користувача. Dashboard повинен бути максимально ергономічним, з інтуїтивно зрозумілим розташуванням елементів, підтримкою темної теми (критично важливо для умов

недостатнього освітлення операторських приміщень металургійного цеху), адаптивністю під різні розміри екранів та можливістю швидкого перемикання між загальним оглядом, детальним трендом окремої секції та історичним аналізом. Інтерфейс має забезпечувати мінімальне когнітивне навантаження на диспетчера в умовах високої інтенсивності роботи.

Розробка веб-додатка здійснюється з дотриманням вимог промислових стандартів до систем реального часу та інформаційної безпеки, що підтверджується рекомендаціями щодо промислових мереж та інформаційних технологій у промисловості [1; 3].

Висновки до розділу 1

У розділі 1 було проведено комплексний аналіз предметної області термомеханічного зміцнення арматури. Встановлено, що стабільність механічних властивостей готової продукції безпосередньо залежить від точного дотримання температурно-швидкісних режимів у багатозонній ТМТ-касеті, а відхилення навіть на кілька градусів або секунд призводить до суттєвого браку.

Огляд існуючих аналогів виявив, що класичні SCADA-системи не забезпечують сучасного веб-доступу та гнучкості, а IoT-платформи недостатньо адаптовані під специфіку металургійного процесу та мають додаткові затримки. Порівняльний аналіз підтвердив відсутність готового рішення, яке б поєднувало високу продуктивність, повну кастомізацію та прийнятну вартість.

Таким чином, на основі проведеного дослідження було чітко сформульовано постановку задачі та обґрунтовано комплекс функціональних і нефункціональних вимог до розроблюваного веб-додатка. Отримані результати створюють необхідну методологічну основу для подальшого проєктування інформаційної моделі, математичного моделювання та практичної реалізації системи в наступних розділах.

Було сформульовано чітку рольову модель системи, яка враховує особливості роботи металургійного цеху. На підприємстві доступ до веб-додатка диспетчеризації мають три основні категорії користувачів.

Перша і найважливіша роль – Диспетчер. Це основний користувач системи, який постійно знаходиться в операторській цеху і відповідає за поточний технологічний процес. Диспетчер повинен у реальному часі бачити графіки температури, тиску води та швидкості прокатки, отримувати миттєві алерти при відхиленнях і мати можливість швидко реагувати на них (наприклад, коригувати тиск води або швидкість лінії). Для цієї ролі надається повний доступ до реал-тайм дашборду, системи алертів та оперативних налаштувань порогових значень.

Друга роль – Технолог. Цей користувач не працює безпосередньо за пультом керування в реальному часі. Його завдання – аналізувати історичні дані за минулі зміни, будувати тренди, порівнювати режими охолодження різних партій і коригувати технологічні карти. Технолог має доступ до всіх історичних даних і аналітичних звітів, але не може змінювати поточні параметри процесу в реальному часі. Такий розподіл прав дозволяє технологу спокійно проводити глибокий аналіз, не відволікаючись на поточні аварійні ситуації.

Третя роль – Адміністратор. До неї належать працівники IT-служби або відповідальні фахівці, які керують обліковими записами, налаштовують підключення нових датчиків, встановлюють права доступу для інших користувачів і виконують загальне адміністрування системи. Адміністратор має доступ до всіх налаштувань, але зазвичай не втручається в технологічний процес.

Такий розподіл ролей і прав доступу є критичним для металургійного підприємства. На заводі одночасно працюють люди з різними обов’язками і рівнем відповідальності. Якщо диспетчер зможе змінювати технологічні налаштування, а технолог – оперативно керувати процесом, це може призвести до хаосу і технологічних порушень. Навпаки, обмеження прав доступу дозволяє кожному спеціалісту виконувати свою роботу ефективно і безпечно. Крім того, на великих підприємствах типу АрселорМіттал чіткий розподіл ролей допомагає

швидко визначити, хто саме відповідає за те чи інше рішення у випадку розслідування інциденту.

Окрім функціональних вимог, були чітко визначені нефункціональні вимоги, які випливають з реальних умов роботи металургійного цеху.

Інтерфейс веб-додатка повинен бути максимально адаптований до умов операторської. У цеху часто погане освітлення, багато пилу і постійний шум. Тому було вирішено використовувати темну кольорову схему (dark theme). Темний фон значно зменшує навантаження на очі диспетчера, особливо під час 12-годинної зміни. Усі графіки, індикатори і кнопки мають бути великими, з високим контрастом і чіткими кольорами. Аварії повинні виділятися яскравим червоним кольором, попередження – жовтим. Шрифти повинні бути великими і добре читабельними навіть з відстані 1,5–2 метри від монітора. Інтерфейс має бути інтуїтивно зрозумілим, щоб диспетчер, який працює втомлений і під тиском, міг миттєво помітити проблему і відреагувати на неї.

На більшості металургійних підприємств комп’ютери в операторських – це не сучасні потужні машини. Часто це промислові ПК 5–8-річної давнини з обмеженими ресурсами процесора і оперативної пам’яті. Тому веб-додаток було спроєктовано як легку Single Page Application (SPA). Уся важка логіка (математична обробка даних, фільтрація шуму, генерація звітів) виконується на бекенді. Фронтенд отримує вже готові дані і лише відображає їх. Такий підхід дозволяє системі стабільно працювати навіть на старих комп’ютерах, не створюючи додаткового навантаження на процесор диспетчерського ПК.

У реальних цехових умовах зв’язок може бути нестабільним. Можливі короткочасні перебої мережі через вібрацію, електромагнітні перешкоди або технічні роботи. Тому веб-додаток повинен автоматично відновлювати WebSocket-з’єднання без необхідності вручну оновлювати сторінку (F5). Користувач не повинен помічати короткочасні перебої – система має самостійно перепідключатися і продовжувати відображення даних. Крім того, додаток

повинен зберігати останній стан (останній графік і список алертів) навіть при тимчасовій втраті зв’язку, щоб диспетчер не втрачав контроль над ситуацією.

Таким чином, при постановці задачі і формулюванні вимог до веб-додатка диспетчеризації було враховано не тільки технічні аспекти, але й реальні умови експлуатації на металургійному підприємстві. Чіткий розподіл ролей, зручний і легкий інтерфейс, адаптований до цехового середовища, а також висока надійність системи є обов’язковими умовами для успішного впровадження розробленого рішення.

РОЗДІЛ 2
ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ ВЕБ-ДОДАТКА ДЛЯ МОНІТОРИНГУ ТА ДИСПЕТЧЕРИЗАЦІЇ ТЕРМОСТАТНОЇ СИСТЕМИ ПРОМИСЛОВОГО ПІДПРИЄМСТВА

2.1. Аналіз бізнес-процесів диспетчеризації та концептуальне проєктування інформаційної моделі системи

Перед початком безпосередньої розробки програмного забезпечення було проведено детальний аналіз бізнес-процесів диспетчеризації температурних режимів системи термомеханічного зміцнення арматури. Такий аналіз є важливим етапом, оскільки дозволяє зрозуміти, як саме відбувається контроль технологічного процесу на підприємстві сьогодні, виявити вузькі місця та сформувати чіткі вимоги до майбутньої системи. Без вивчення реальних бізнес-процесів існує ризик створити продукт, який технічно працюватиме, але не вирішуватиме щоденні завдання диспетчерів, технологів і керівництва цеху. Тому спочатку було досліджено поточну модель роботи «як є» (AS-IS), потім сформовано цільову модель «як має бути» (TO-BE), а на їхній основі побудовано концептуальну інформаційну модель системи за допомогою мови UML.

На даний момент диспетчеризація процесу термомеханічного зміцнення арматури на більшості прокатних станів здійснюється за допомогою застарілих локальних SCADA-систем. Диспетчер працює за одним стаціонарним комп’ютером, який фізично знаходиться в операторській кабінці безпосередньо біля лінії прокатки. Вся телеметрія від датчиків температури, тиску води та швидкості прокатки надходить виключно на цей комп’ютер. Диспетчер постійно спостерігає за графіками і реагує на сигнали, що з’являються на пульті керування. Якщо потрібно переглянути дані за попередню зміну, технолог або начальник

зміни змушені особисто приходити в операторську і просити диспетчера вивантажити звіт на флешку або роздрукувати його.

Така організація процесу має ряд суттєвих недоліків. По-перше, вся інформація прив’язана до одного робочого місця, тому доступ до даних мають лише ті працівники, які фізично знаходяться в операторській. По-друге, історичні дані зберігаються локально, і при поломці комп’ютера або проблемах з жорстким диском інформація може бути втрачена. По-третє, аварійні сигнали часто обмежуються лише звуковим або світловим сповіщенням, без фіксації точного часу виникнення відхилення та часу реакції диспетчера. Як зазначається у джерелах, сучасні SCADA-системи на багатьох підприємствах уже не відповідають вимогам Industry 4.0 через низьку доступність даних і відсутність централізованого зберігання [4].

У результаті диспетчер часто працює в умовах підвищеного навантаження, а технолог та керівництво не мають оперативного доступу до інформації. Це призводить до затримок у прийнятті рішень, збільшення ризику людського фактора та вищого рівня браку продукції через несвоєчасне реагування на відхилення температурного режиму.

Після впровадження розробленого веб-додатка диспетчеризації бізнес-процеси контролю за охолодженням арматури кардинально змінюються. Система переходить на сучасну клієнт-серверну архітектуру. Усі обчислення, обробка телеметрії та зберігання даних виконуються на виділеному сервері. Диспетчер, технолог, начальник зміни та інші уповноважені фахівці отримують доступ до інформації через звичайний веб-браузер з будь-якого комп’ютера або планшета, підключеного до заводської мережі.

Диспетчер продовжує працювати в операторській, але тепер він користується сучасним інтерактивним Dashboard, який оновлюється в реальному часі через WebSocket. При виникненні відхилення температури або тиску води сповіщення миттєво з’являється на екрані. Технолог може зі свого робочого місця в іншому корпусі відкрити дашборд, переглянути детальні графіки за будь-який

період, порівняти режими різних змін і підготувати рекомендації щодо оптимізації процесу. Начальник зміни також отримує доступ до актуальних даних без необхідності йти в операторську.

Вся історія телеметрії зберігається в централізованій базі даних PostgreSQL. Кожна секунда показників фіксується з точним часом. Це дозволяє швидко проводити аналіз, будувати звіти і виявляти приховані закономірності. Автоматичні алерти через WebSocket значно підвищують оперативність реагування. Як зазначається в джерелах, перехід до веб-орієнтованих систем моніторингу на базі IoT-платформ і протоколів Industry 4.0 дозволяє суттєво підвищити доступність даних і швидкість прийняття рішень [1; 2; 3].

Такий перехід від локальної моделі AS-IS до веб-орієнтованої моделі TO-BE знижує залежність від одного робочого місця, зменшує вплив людського фактора, забезпечує прозорість даних для всіх зацікавлених фахівців і створює умови для глибокого аналізу технологічних режимів.

Перехід до концептуального проєктування інформаційної моделі

На основі проведеного аналізу моделей AS-IS і TO-BE були сформульовані чіткі вимоги до майбутньої системи. Для формалізації цих вимог і візуального опису взаємодії компонентів було вирішено використати мову моделювання UML. Такий підхід дозволяє чітко визначити, хто і як взаємодіє з системою, а також описати основні сутності даних і їх зв’язки між собою.

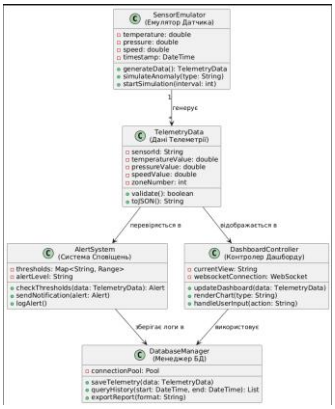


Рис. 2.1. UML-діаграма прецедентів системи

На діаграмі прецедентів чітко визначено рольову модель системи. Основними акторами виступають три ролі:

Диспетчер – основний користувач системи. Він виконує прецеденти «Авторизація в системі», «Перегляд реал-тайм дашборду», «Отримання та підтвердження алертів», «Моніторинг поточних параметрів» і «Коригування порогових значень».

Технолог – має доступ до історичних даних. Виконує прецеденти «Перегляд архівних даних», «Побудова трендів і звітів», «Аналіз технологічних режимів за минулі зміни».

Адміністратор – керує системою. Виконує прецеденти «Управління обліковими записами користувачів», «Налаштування підключення нових датчиків», «Перегляд журналу аудиту» і «Керування правами доступу».

Такий розподіл ролей дозволяє чітко розмежувати права доступу і забезпечити безпеку системи. Наприклад, диспетчер має право оперативно реагувати на алерти, але не може змінювати глобальні налаштування системи, а технолог може аналізувати історичні дані, але не має доступу до реального часу керування.

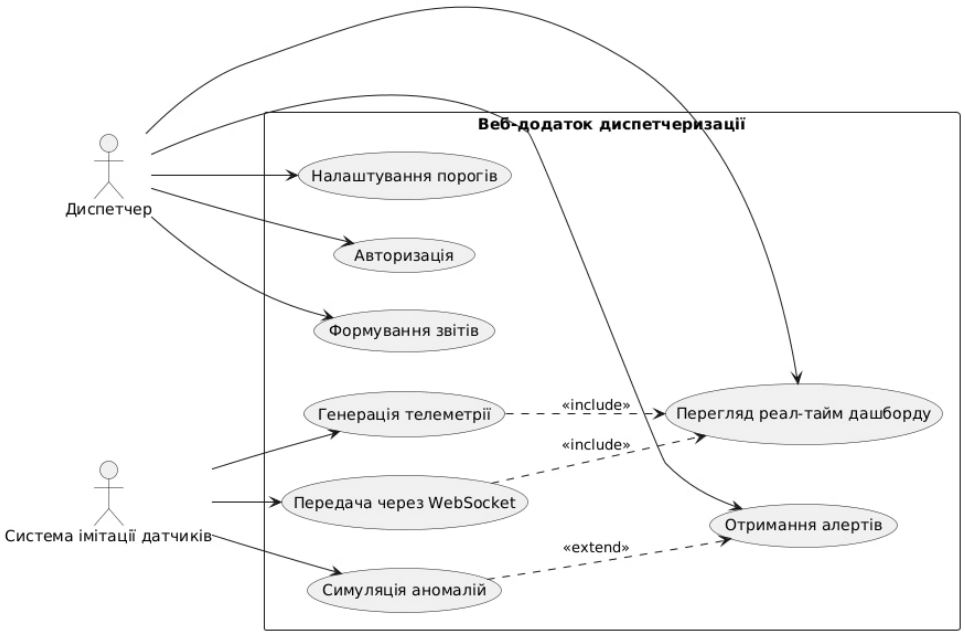


Рис. 2.2. UML-діаграма класів

Для опису статичної структури даних була побудована діаграма класів. Основними класами є:

User (Користувач) – містить інформацію про всіх користувачів системи (id, username, password_hash, role, created_at). Кожен користувач може мати одну з ролей: диспетчер, технолог або адміністратор.

Sensor (Датчик) – довідник датчиків (id, zone_number, type, min_threshold, max_threshold, is_active). Один датчик може генерувати багато записів телеметрії.

TelemetryLog (Журнал телеметрії) – основна таблиця для зберігання історичних даних (id, sensor_id, temperature, pressure, speed, timestamp). Кожен запис пов’язаний з конкретним датчиком.

Alert (Журнал аварій) – фіксує всі відхилення (id, sensor_id, alert_level, description, timestamp, is_resolved, resolved_at). Кожен алерт пов’язаний з певним датчиком і може бути пов’язаний з користувачем, який підтвердив його.

Між класами встановлено чіткі зв’язки: один датчик генерує багато записів телеметрії і може створювати багато аварійних повідомлень. Кожен користувач може переглядати дані від усіх датчиків, але тільки диспетчер може підтверджувати алерти в реальному часі.

Таким чином, проведений аналіз бізнес-процесів і побудована концептуальна інформаційна модель створюють надійну основу для подальшої практичної реалізації веб-додатка диспетчеризації.

2.2. Математичне моделювання алгоритмів збору та обробки телеметричних даних (імітація показників датчиків)

У зв’язку з неможливістю прямого підключення до реального промислового обладнання на етапі дипломного проєктування було розроблено математичну модель імітації телеметричних даних. Модель забезпечує

максимально реалістичне відтворення поведінки датчиків температури, тиску та швидкості прокатки в умовах багатозонної ТМТ-касети.

Для опису процесу охолодження металу в ТМТ-касеті застосовано закон Ньютона–Ріхмана. Математична модель зміни температури поверхні записується у вигляді диференціального рівняння:

$$\frac{dT}{dt} = -k\,(T - T_{env})$$

де T – поточна температура поверхні металу, °C;

t – час, с;

k – коефіцієнт тепловіддачі (0,12 ÷ 0,28 $с^{-1}$);

T_{env} – температура охолоджувальної води, °C.

Оскільки в умовах реального цеху на показання датчиків впливають вібрація, електромагнітні перешкоди та запиленість, до ідеальної температури додається стохастична компонента – білий шум за нормальним розподілом:

$$T_{noisy(t)} = T_{ideal(t)} + N(0, \sigma^2)$$

де N(0, σ^2) – нормально розподілена випадкова величина з середньоквадратичним відхиленням σ = 3 ÷ 8 °C.

Для згладжування зашумлених показників перед виведенням на графіки Dashboard застосовується метод ковзного середнього. Формула розрахунку має вигляд:

$$SMA_t = \left(\frac{1}{n}\right) \Sigma_{\{i=0\}^{\{n-1\}T}_{noisy(t-i)}}$$

Розроблена математична модель враховує як детерміновану складову процесу термомеханічного зміцнення [7], так і стохастичні фактори реального виробництва. Вона забезпечує високий ступінь наближення імітаційних даних до фактичних показників датчиків на прокатному стані.

2.3. Проєктування архітектури бази даних для збереження історії показників (ER-діаграма)

Для ефективного зберігання високочастотної телеметрії, що надходить від датчиків системи термомеханічного зміцнення арматури, було розроблено архітектуру бази даних. Вибір реляційної моделі даних і конкретної системи управління базами даних здійснювався з урахуванням особливостей промислового виробництва, де важливі не тільки швидкість запису даних, але й їх цілісність, надійність та можливість швидкого виконання аналітичних запитів. Було встановлено, що для даного проєкту оптимальним рішенням є використання реляційної СУБД. Реляційна модель дозволяє чітко визначити зв'язки між сутностями, забезпечити цілісність даних на рівні обмежень (constraints) і виконувати складні аналітичні запити, необхідні для побудови трендів і звітів.

Як основна СУБД для промислового розгортання було обрано PostgreSQL. Цей вибір обґрунтовується тим, що PostgreSQL повністю підтримує ACID-транзакції (Atomicity, Consistency, Isolation, Durability), що є критичним для підприємства, де кожне відхилення температури або тиску може призвести до браку продукції вартістю в десятки тисяч гривень. ACID-транзакції гарантують, що дані або повністю записуються, або не записуються взагалі, навіть у разі збою обладнання чи мережі. Крім того, PostgreSQL добре масштабується, має потужні механізми індексації та вбудовану підтримку часових рядів, що ідеально підходить для зберігання мільйонів записів телеметрії. Для етапу розробки та локального тестування було використано SQLite. Ця СУБД не вимагає встановлення окремого сервера, є повністю портативною і дозволяє швидко запускати проєкт на будь-якому комп'ютері. Такий гібридний підхід (PostgreSQL у продакшні та SQLite під час розробки) є поширеним у сучасних IT-проєктах для промисловості, оскільки поєднує надійність і зручність [3].

Порівняно з NoSQL-рішеннями (наприклад, MongoDB), реляційна модель виявилася більш придатною. NoSQL-бази даних краще підходять для

неструктурованих даних або дуже високих навантажень без жорстких зв’язків, проте в даному випадку потрібна чітка структура: кожен запис телеметрії повинен бути строго пов’язаний з конкретним датчиком, а кожен алерт – з певним користувачем і часом. Реляційна модель з нормалізацією дозволяє уникнути дублювання даних і забезпечити високу швидкість аналітичних запитів, що особливо важливо для Dashboard, де графіки повинні будуватися за лічені мілісекунди.

Було вирішено привести базу даних до третьої нормальної форми (3NF). Це дозволило усунути дублювання даних, зменшити обсяг зберігання та уникнути аномалій оновлення. Розроблена ER-діаграма складається з чотирьох основних взаємопов’язаних сутностей (таблиць): users, sensors, telemetry_logs та alerts. Зв’язки між таблицями реалізовані через зовнішні ключі (foreign keys), що забезпечує цілісність даних на рівні бази.

Таблиця users призначена для зберігання облікових записів усіх користувачів системи. Вона містить поля: id (SERIAL PRIMARY KEY), username (VARCHAR(50) UNIQUE NOT NULL), password_hash (VARCHAR(255) NOT NULL), role (ENUM('dispatcher', 'technologist', 'admin')), created_at (TIMESTAMP DEFAULT CURRENT_TIMESTAMP) та last_login (TIMESTAMP). Паролі зберігаються виключно у хешованому вигляді для забезпечення безпеки. Такий підхід є обов’язковим у сучасних промислових системах, оскільки несанкціонований доступ до панелі диспетчера може призвести до серйозних технологічних порушень. Ролі користувачів чітко розмежовують права доступу. Логування дій диспетчера (через пов’язані таблиці) відповідає вимогам сучасних SCADA-систем до аудиту та відповідальності персоналу [4].

Таблиця sensors є довідником датчиків лінії ТМЗ. Вона містить поля: id (SERIAL PRIMARY KEY), zone_number (INTEGER NOT NULL), type (VARCHAR(30)), min_threshold (NUMERIC(6,2)), max_threshold (NUMERIC(6,2)), is_active (BOOLEAN DEFAULT true). Ця таблиця дозволяє динамічно налаштовувати порогові значення для алертів без необхідності

змінювати код програми. Наприклад, технолог може в інтерфейсі змінити межі температури для певної зони, і система одразу почне працювати за новими параметрами. Такий підхід значно підвищує гнучкість системи.

Таблиця telemetry_logs є основною і найбільш завантаженою таблицею системи. Вона призначена для зберігання високочастотної телеметрії. Структура включає: id (BIGSERIAL PRIMARY KEY), sensor_id (INTEGER REFERENCES sensors(id)), temperature (NUMERIC(6,2)), pressure (NUMERIC(6,2)), speed (NUMERIC(6,2)), timestamp (TIMESTAMP DEFAULT CURRENT_TIMESTAMP). Тип BIGSERIAL для первинного ключа обрано через очікуваний великий обсяг даних – мільйони записів за кілька місяців роботи. Тип NUMERIC(6,2) забезпечує необхідну точність для температур і тиску (два знаки після коми). Ця таблиця є джерелом даних для всіх графіків Dashboard.

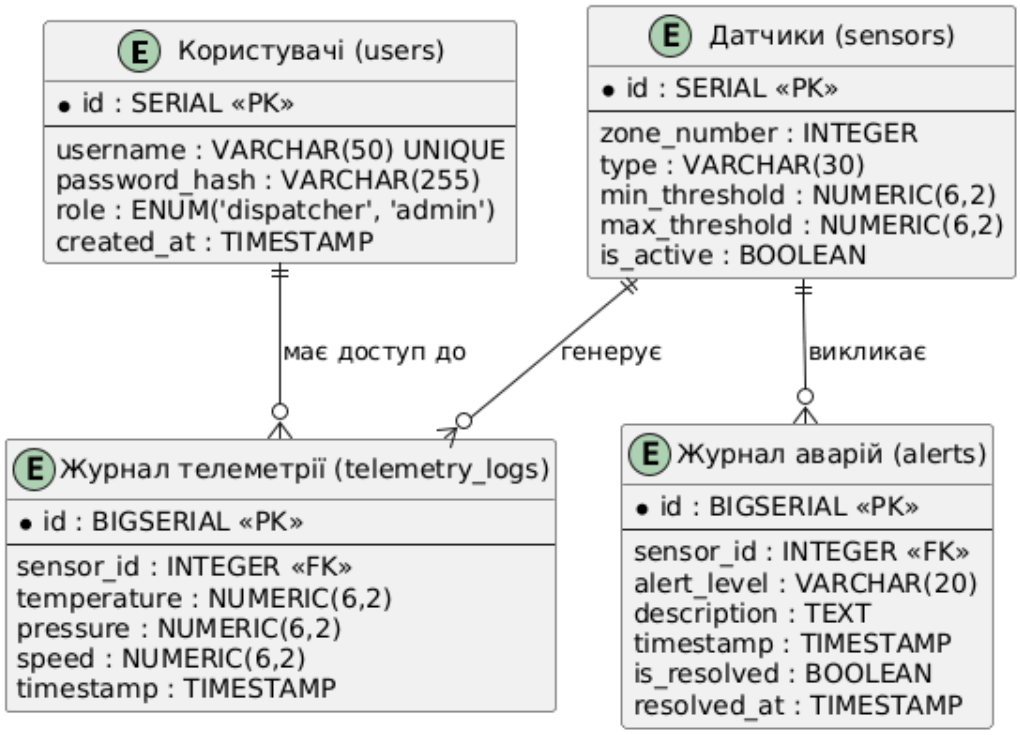
Таблиця alerts призначена для фіксації всіх технологічних порушень. Поля: id (BIGSERIAL PRIMARY KEY), sensor_id (INTEGER REFERENCES sensors(id)), alert_level (VARCHAR(20)), description (TEXT), timestamp (TIMESTAMP), is_resolved (BOOLEAN DEFAULT false), resolved_at (TIMESTAMP). Поля is_resolved та resolved_at дозволяють точно відстежувати, коли саме виникла аварійна ситуація і як швидко диспетчер на неї відреагував. Це дає керівництву можливість об’єктивно оцінювати роботу персоналу.

Для забезпечення швидкої роботи Dashboard було спроєктовано систему індексації. На таблиці telemetry_logs створено композитні B-Tree індекси по полях timestamp та sensor_id. Без таких індексів запити на вибірку даних за останню добу або тиждень могли б «покласти» базу даних через повне сканування таблиці. Індексація значно прискорює вибірку даних для побудови графіків і формування звітів. Крім того, були створені індекси на зовнішні ключі (sensor_id у таблицях telemetry_logs та alerts), що прискорює операції JOIN.

Оптимізація бази даних є особливо важливою в умовах промислових мереж Industry 4.0, де обсяг даних постійно зростає [1]. Правильно спроєктована

індексація дозволяє системі працювати стабільно навіть при одночасному підключенні кількох диспетчерів і технологів.

Таким чином, розроблена архітектура бази даних повністю відповідає вимогам реального промислового виробництва, забезпечує надійне зберігання даних,



швидкий доступ до інформації та високу безпеку. Спроєктована структура є масштабована і може бути легко розширена при додаванні нових ліній прокатки або датчиків.

Рис. 2.3. PlantUML-код для ER-діаграми

У підрозділі 2.3 була спроектована архітектура бази даних для зберігання телеметричних даних та алертів. Було обрано реляційну модель з нормалізацією до третьої нормальної форми та створено індекси за часом і зовнішніми ключами. Це дозволяє забезпечити цілісність інформації та швидкий доступ до даних при формуванні графіків дашборду. Для розробки використано SQLite, а для промислового впровадження рекомендовано PostgreSQL, що відповідає вимогам надійності систем моніторингу на металургійних підприємствах.

Висновки до розділу 2

У розділі 2 було здійснено концептуальне проєктування системи. Було побудовано інформаційну модель за допомогою UML-діаграм прецедентів та класів, що дозволило формалізувати вимоги та структуру майбутнього програмного забезпечення.

Розроблено математичну модель імітації телеметричних даних, яка поєднує детерміновані закони теплопередачі з стохастичними компонентами реального виробництва. Запропоновані алгоритми згладжування забезпечують якісну підготовку даних для візуалізації.

На завершення спроєктовано архітектуру реляційної бази даних, яка гарантує ефективне зберігання, швидкий пошук та цілісність високочастотної телеметрії. Сукупність отриманих результатів створює повноцінну методологічну та технічну основу для переходу до практичної реалізації програмного забезпечення в розділі 3.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВЕБ-ДОДАТКА МОНІТОРИНГУ ТА ДИСПЕТЧЕРИЗАЦІЇ ПРОЦЕСУ ТЕРМОМЕХАНІЧНОГО ЗМІЦНЕННЯ АРМАТУРИ

3.1. Аналіз вимог та постановка задачі розробки програмного забезпечення

У сучасних умовах цифровізації промислового виробництва все більшої актуальності набуває впровадження систем моніторингу технологічних процесів, побудованих на принципах концепції Industry 4.0 [1, 3]. Одним із ключових напрямів такої цифровізації є створення інтегрованих веб-орієнтованих рішень, здатних забезпечити оперативний контроль параметрів виробництва в реальному часі без прив’язки до стаціонарних робочих місць оператора. У рамках даної роботи було розроблено веб-додаток для моніторингу та диспетчеризації процесу термомеханічного зміцнення арматури, що дозволяє здійснювати безперервний контроль температурних режимів у зонах ТМТ-касети.Загальна мета розробки полягала у створенні сучасного програмного комплексу, який забезпечує збір, обробку, візуалізацію та зберігання телеметричних даних про температурні режими в п’яти зонах охолодження арматури. Особливістю розроблюваного рішення є поєднання реалістичної фізичної моделі поведінки температури з інтерактивним веб-інтерфейсом, орієнтованим на практичне використання диспетчерським та технологічним персоналом промислового підприємства. Реалізація такої системи дозволяє підвищити стабільність технологічного процесу та забезпечити відповідність продукції вимогам ДСТУ 3760:2019 щодо механічних характеристик арматури [21].

На основі аналізу особливостей технологічного процесу та потреб потенційних користувачів було сформульовано наступні функціональні вимоги до програмного забезпечення:

Система повинна забезпечувати безперервний збір телеметричних даних з п’яти незалежних зон ТМТ-касети. До складу вимірюваних параметрів входять температура металу, тиск води в системі охолодження та швидкість прокатки. З метою наближення до реальних умов експлуатації в серверній частині застосовано емуляцію роботи датчиків, побудовану на основі закону охолодження Ньютона з додаванням гаусівського шуму та застосуванням алгоритму ковзного середнього для згладжування показників. Такий підхід дозволяє імітувати як фізичні процеси охолодження, так і похибки реальних вимірювальних приладів.

Важливою функціональною вимогою є відображення даних у реальному часі на веб-інтерфейсі. Для цього реалізовано механізм асинхронної передачі даних за допомогою протоколу WebSocket, що забезпечує оновлення показників без перезавантаження сторінки. На головній панелі дашборду відображаються ключові показники ефективності (KPI): середня температура, середній тиск води, середня швидкість прокатки та кількість активних алертів. Окремо реалізовано інтерактивний графік, на якому одночасно відображається динаміка трьох параметрів з використанням двох вертикальних осей.

Система повинна автоматично виявляти відхилення температури від заданих порогових значень у кожній зоні та формувати відповідні алерти. Для кожного алерту фіксується час виникнення, рівень критичності (warning або critical) та детальний опис події. Алерти зберігаються у базі даних і доступні для перегляду як у режимі реального часу, так і в історії. Для користувачів з роллю «Диспетчер» передбачено можливість підтвердження (квітування) алертів, що фіксує факт обробки події відповідальною особою.

Додатковою функціональною вимогою є можливість оперативного керування пороговими значеннями температури окремо для кожної зони безпосередньо через веб-інтерфейс. Зміна порогів одразу впливає на логіку формування алертів без необхідності перезапуску серверної частини. Також

реалізовано функцію експорту історії алертів у форматі CSV-файлу для подальшого аналізу та формування звітності.

До розроблюваного програмного забезпечення було висунуто низку нефункціональних вимог, що випливають із особливостей промислової експлуатації.

По-перше, система повинна забезпечувати високу продуктивність та низьку затримку передачі даних. Використання WebSocket-протоколу замість традиційного HTTP-polling дозволяє досягти оновлення даних з періодичністю менше однієї секунди, що критично важливо для оперативного реагування на технологічні відхилення.

По-друге, програмне забезпечення повинно бути надійним та відмовостійким. У клієнтській частині реалізовано механізм автоматичного перепідключення до WebSocket-сервера у разі втрати зв’язку, що забезпечує відновлення функціонування системи без втручання користувача. Зберігання даних у локальній базі даних SQLite гарантує збереження телеметрії та історії алертів навіть при тимчасовій недоступності клієнтських робочих місць.

По-третє, система має бути простою у розгортанні та подальшій підтримці. Застосування контейнеризації за допомогою Docker та використання мінімальної кількості залежностей дозволяє швидко розгорнути рішення як у тестовому, так і у промисловому середовищі.

По-четверте, інтерфейс користувача повинен бути інтуїтивно зрозумілим та відповідати сучасним вимогам до промислових дашбордів. Темна колірна схема, чітке візуальне розділення функціональних блоків, використання кольорового кодування станів (зелений – норма, червоний – критичне відхилення) та генерація звукових сигналів при критичних алертах сприяють зниженню когнітивного навантаження на оператора.

На відміну від застарілих локальних SCADA-систем, які часто характеризуються високою вартістю ліцензування, складністю масштабування та прив’язкою до конкретних робочих місць [4], розроблений веб-додаток

забезпечує доступ до даних з будь-якого пристрою в межах локальної мережі підприємства через звичайний веб-браузер. Це відповідає сучасним тенденціям цифровізації промислових мереж та концепції Industry 4.0 [1, 3].

Таким чином, сформульовані вимоги лягли в основу архітектури та реалізації програмного комплексу, що поєднує в собі можливості сучасної системи моніторингу з високою доступністю, простотою експлуатації та відповідністю вимогам до якості продукції згідно з ДСТУ 3760:2019 [21].

3.2. Вибір архітектури та технологій розробки

При проектуванні програмного забезпечення веб-додатка для моніторингу температурних режимів у процесі термомеханічного зміцнення арматури було обрано клієнт-серверну архітектуру з чітким розділенням відповідальності між серверною та клієнтською частинами. Такий підхід дозволяє забезпечити незалежний розвиток, масштабування та підтримку компонентів системи, а також спрощує інтеграцію з іншими інформаційними системами підприємства. Серверна частина відповідає за збір, обробку та зберігання даних, тоді як клієнтська частина забезпечує їх візуалізацію та взаємодію з користувачем.

В якості основної мови програмування серверної частини було обрано Python у поєднанні з асинхронним фреймворком FastAPI. Висока продуктивність FastAPI, заснована на використанні асинхронного runtime-оточення Starlette та бібліотеки Pydantic для валідації даних, дозволяє ефективно обробляти велику кількість одночасних підключень WebSocket, що є критичним для систем реального часу в металургійному виробництві [8]. У розробленому рішенні FastAPI використовується для організації WebSocket-ендпоінту /ws/telemetry, обробки REST-запитів на отримання історії та підтвердження алертів, а також для забезпечення CORS-політики при взаємодії з клієнтською частиною.

Для передачі даних у реальному часі було обрано протокол WebSocket відповідно до специфікації RFC 6455 [11]. На відміну від традиційного HTTP-polling, WebSocket забезпечує повнодуплексний канал зв'язку з мінімальною затримкою та низьким накладними витратами на встановлення з'єднання. У реалізованому коді WebSocket-сервер підтримує список активних клієнтських підключень та здійснює широкомовну розсилку (broadcast) телеметричних даних усім підключеним клієнтам з періодичністю приблизно 0,85 секунди. Крім того, реалізовано механізм автоматичного перепідключення на клієнтській стороні при втраті зв'язку, що підвищує надійність системи в умовах нестабільної промислової мережі.

Щодо системи управління базами даних, було прийнято рішення використовувати SQLite замість PostgreSQL. Хоча PostgreSQL традиційно вважається більш потужним рішенням для високонавантажених систем [10], у рамках даного проєкту SQLite було обрано з огляду на простоту розгортання, відсутність необхідності в окремому сервері бази даних та достатню продуктивність для сценарію з одним записуючим потоком (серверна частина) та кількома читаючими клієнтами. У коді SQLite використовується для зберігання таблиць telemetry_logs та alerts з підтримкою транзакцій та індексації за часом.

Для клієнтської частини було свідомо відхилено використання JavaScript-фреймворків, зокрема React [9], на користь чистого HTML5, CSS (Tailwind CSS через CDN) та vanilla JavaScript. Такий підхід дозволив значно зменшити розмір кінцевого застосунку, прискорити час завантаження сторінки та спростити процес розгортання без необхідності використання складних інструментів збірки (Webpack, Vite тощо). Уся бізнес-логіка, включаючи обробку WebSocket-повідомлень, оновлення DOM-дерева, формування алертів та експорт даних у CSV, реалізована безпосередньо у файлі index.html.

Нарешті, для забезпечення портативності та спрощення процесу розгортання системи було застосовано технологію контейнеризації Docker [22]. Використання docker-compose.yml дозволяє одночасно запускати серверну

частину (FastAPI) та статичний веб-сервер для клієнтської части у ізольованих контейнерах з чітко визначеними залежностями. Це значно спрощує як локальну розробку, так і впровадження рішення на промислових серверах підприємства.

Таким чином, обраний технологічний стек забезпечує оптимальне співвідношення продуктивності, простоти розробки та надійності для побудови системи моніторингу реального часу в умовах металургійного виробництва.

3.3 Розробка серверної і клієнтської частини

Серверна частина веб-додатка розроблена на основі асинхронного фреймворку FastAPI та призначена для збору, обробки, зберігання телеметричних даних та формування алертів у реальному часі. Архітектура серверної частини побудована за принципом єдиного відповідального модуля, що дозволяє забезпечити високу продуктивність при обробці великої кількості одночасних WebSocket-підключень.

Основним компонентом серверної частини є екземпляр класу FastAPI, створений у файлі backend/main.py. При ініціалізації застосунку було налаштовано middleware CORSMiddleware для забезпечення крос-доменних запитів з клієнтської частини. Серверна частина використовує асинхронну модель виконання на основі модуля asyncio, що дозволяє ефективно обробляти I/O-операції без блокування основного потоку виконання [8].

Центральним елементом взаємодії з клієнтами є WebSocket-ендпоінт /ws/telemetry. При підключенні клієнта його об’єкт WebSocket додається до глобального списку active_connections. Усередині циклу обробки відбувається формування пакетів телеметрії від усіх п’яти емуляторів датчиків з подальшою широкомовною розсилкою даних усім активним клієнтам. Для запуску застосунку використовується ASGI-сервер **Uvicorn**, який забезпечує високу продуктивність та підтримку WebSocket-протоколу [23].

```python
@app.websocket("/ws/telemetry")
async def websocket_endpoint(websocket: WebSocket):
    await websocket.accept()
    active_connections.append(websocket)

    try:
        while True:
            batch = []
            for zone_id, emulator in emulators.items():
                reading = emulator.step()
                batch.append(reading)

                t = reading["temperature"]
                th = thresholds[zone_id]

                # Логіка алертів з дедуплікацією
                is_out_of_range = t < th["min"] or t > th["max"]
                if is_out_of_range and not active_alerts[zone_id]:
                    level = "critical" if (t < th["min"] - 15 or t > th["max"] + 15) else "w
                    save_alert(zone_id, level, f"Температура зони {zone_id}: {t}°C")
                    active_alerts[zone_id] = True
                elif not is_out_of_range and active_alerts[zone_id]:
                    active_alerts[zone_id] = False

            # Розсилка даних клієнтам
            message = json.dumps({"type": "telemetry", "data": batch})
            for conn in active_connections[:]:
                try:
                    await conn.send_text(message)
                except:
                    active_connections.remove(conn)

            if random.random() < 0.6:
                save_telemetry_batch(batch)

            await asyncio.sleep(0.9)
    except WebSocketDisconnect:
        if websocket in active_connections:
            active_connections.remove(websocket)
```

Рис. 3.1. WebSocket-ендпоінт та основна логіка обробки телеметрії

Для імітації роботи реальних датчиків температури у кожній зоні ТМТ-касети було розроблено клас SensorEmulator. Клас реалізує фізичну модель охолодження на основі диференціального рівняння закону Ньютона з урахуванням випадкового гаусівського шуму та алгоритму ковзного середнього для згладжування показників.

У конструкторі класу задаються початкові значення температури, коефіцієнт тепловіддачі k та параметри шуму. Метод step() на кожній ітерації обчислює нове значення температури за формулою:

$$T_{new} = T_{old} - k \cdot (T_{old} - T_{water}) \cdot dt + noise$$

Отримане значення пропускається через буфер розміром 5 елементів, після чого обчислюється середнє арифметичне. Окрім температури, метод генерує значення тиску води та швидкості прокатки з додаванням випадкового

відхилення. Такий підхід дозволяє отримувати реалістичні дані, близькі до показників реальних промислових датчиків.

У файлі main.py створюється словник emulators, що містить п’ять екземплярів класу SensorEmulator (по одному на кожну зону). Генерація даних відбувається у циклі WebSocket-обробника з періодичністю приблизно 0,85 секунди.

```python
class SensorEmulator:
    def __init__(self, zone_id: int):
        self.zone_id = zone_id
        base_temp = 535 + (zone_id * 6)
        self.target_temp = base_temp + random.uniform(-12, 12)
        self.T = self.target_temp + random.uniform(-20, 20)
        self.noise_sigma = 4.2
        self.buffer = []
        self.buffer_size = 5
        self.change_speed = 0.28

    def step(self) -> dict:
        error = self.target_temp - self.T
        drift = error * 0.011
        noise = random.gauss(0, self.noise_sigma)

        self.T += (drift + noise) * self.change_speed

        # Згладжування
        self.buffer.append(self.T)
        if len(self.buffer) > self.buffer_size:
            self.buffer.pop(0)
        smoothed = sum(self.buffer) / len(self.buffer)

        pressure = round(17.9 + random.gauss(0, 0.9), 1)
        speed = round(13.7 + random.gauss(0, 0.55), 1)

        return {
            "zone": self.zone_id,
            "temperature": round(max(420, min(680, smoothed)), 1),
            "pressure": max(12.8, min(26.0, pressure)),
            "speed": max(9.8, min(17.2, speed)),
            "timestamp": datetime.now().isoformat(timespec='seconds')
        }
```

Рис. 3.2 Клас емулятора датчиків

Для зберігання історичних даних телеметрії та алертів у серверній частині використано легку реляційну базу даних **SQLite**. Вибір SQLite обґрунтовано простотою розгортання та достатньою продуктивністю для сценарію з одним записуючим потоком [10].

У файлі backend/main.py реалізовано функцію init_db(), яка створює дві таблиці:

telemetry_logs – для зберігання історичних показників (поля: id, sensor_id, temperature, pressure, speed, timestamp);

alerts – для зберігання подій відхилень (поля: id, sensor_id, alert_level, description, timestamp, is_resolved, resolved_at, resolved_by).

Збереження даних здійснюється у функціях save_telemetry_batch() та save_alert(). При отриманні пакету телеметрії кожне значення записується в таблицю telemetry_logs. У разі виходу температури за межі порогових значень автоматично викликається функція save_alert(), яка фіксує подію з відповідним рівнем критичності.

Для зберігання історичних даних використано SQLite. Нижче наведено реалізацію функцій ініціалізації бази даних та збереження телеметрії й алертів.

```
def init_db():
    conn = sqlite3.connect(DB_PATH)
    c = conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS telemetry_logs (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        sensor_id INTEGER,
        temperature REAL,
        pressure REAL,
        speed REAL,
        timestamp TEXT
    )''')
    c.execute('''CREATE TABLE IF NOT EXISTS alerts (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        sensor_id INTEGER,
        alert_level TEXT,
        description TEXT,
        timestamp TEXT,
        is_resolved INTEGER DEFAULT 0
    )''')
    conn.commit()
    conn.close()

def save_telemetry_batch(batch):
    conn = sqlite3.connect(DB_PATH)
    c = conn.cursor()
    for r in batch:
        c.execute(
            "INSERT INTO telemetry_logs (sensor_id, temperature, pressure, speed, timestamp
            (r["zone"], r["temperature"], r["pressure"], r["speed"], r["timestamp"])
        )
    conn.commit()
    conn.close()

def save_alert(zone_id: int, level: str, description: str):
    conn = sqlite3.connect(DB_PATH)
    c = conn.cursor()
    c.execute(
        "INSERT INTO alerts (sensor_id, alert_level, description, timestamp) VALUES (?, ?,
        (zone_id, level, description, datetime.now().isoformat(timespec='seconds'))
    )
    conn.commit()
    conn.close()
```

Рис. 3.3 Функції роботи з базою даних

Зважаючи на те, що WebSocket-протокол є повнодуплексним каналом зв’язку, до питань безпеки з’єднань було приділено особливу увагу. У розробленому рішенні реалізовано базовий захист на рівні CORS-політики та обмеження доступу до ендпоінтів. Для підвищення рівня захищеності

WebSocket-з’єднань рекомендовано дотримуватися рекомендацій OWASP WebSocket Security Cheat Sheet, зокрема використання захищеного протоколу wss:// у промисловому середовищі, перевірку походження запитів та обмеження максимального розміру повідомлень [16].

У поточній реалізації автентифікація та авторизація користувачів здійснюється на клієнтській стороні з використанням localStorage. На серверній стороні додаткового рівня автентифікації WebSocket-підключень не передбачено, що є прийнятним для внутрішньомережевого використання в межах промислового підприємства. У майбутньому планується впровадження токен-автентифікації (JWT) при встановленні WebSocket-з’єднання.

Таким чином, серверна частина забезпечує надійну генерацію реалістичних телеметричних даних, їх асинхронну обробку та зберігання з використанням сучасних технологій, що відповідає вимогам до систем моніторингу реального часу в металургійній галузі.

Клієнтська частина веб-додатка реалізована у вигляді односторінкового застосунку (Single Page Application) без використання тяжких JavaScript-фреймворків. Такий підхід було обрано з метою забезпечення максимальної швидкості завантаження інтерфейсу та спрощення процесу розгортання в умовах промислового підприємства. Уся логіка відображення даних, обробки подій користувача та взаємодії з сервером зосереджена у єдиному файлі frontend/index.html.

На відміну від компонентного підходу, реалізованого у бібліотеці React [9], у даному проєкті застосовано архітектуру на основі чистого JavaScript (vanilla JS). Це дозволило уникнути необхідності використання інструментів збірки та зменшити розмір кінцевого застосунку. Стан застосунку зберігається у глобальних змінних JavaScript та в localStorage браузера. Такий підхід забезпечує простоту підтримки коду та високу швидкість відгуку інтерфейсу навіть на застарілому обладнанні, яке часто використовується на виробництві.

```
function connectWebSocket() {
    ws = new WebSocket('ws://localhost:8000/ws/telemetry');

    ws.onopen = () => {
        document.getElementById('connection-dot').className = 'w-2.5 h-2.5 rounded-full bg-e
        document.getElementById('connection-text').innerText = 'Підключено • WebSocket';
    };

    ws.onmessage = (event) => {
        try {
            const msg = JSON.parse(event.data);
            if (msg.type === 'telemetry') handleTelemetry(msg.data);
        } catch (e) {
            console.error("Помилка парсингу WebSocket:", e);
        }
    };

    ws.onclose = () => {
        document.getElementById('connection-dot').className = 'w-2.5 h-2.5 rounded-full bg-r
        document.getElementById('connection-text').innerText = '3\'єднання втрачено. Перепід
        setTimeout(connectWebSocket, 3000);
    };
}
```

Рис.3.4. Підключення та обробка WebSocket

```
function handleTelemetry(data) {
    if (!data || data.length === 0) return;

    // Оновлення KPI-карток
    updateKPICards(data);

    // Оновлення статусу зон
    updateZoneStatus(data);

    // Оновлення графіка Chart.js
    if (mainChart) {
        const now = new Date().toLocaleTimeString('uk-UA', { hour: '2-digit', minute: '2-dig

        mainChart.data.labels.push(now);
        if (mainChart.data.labels.length > 40) mainChart.data.labels.shift();

        // Оновлення даних для кожної зони (приклад для зони 1)
        mainChart.data.datasets[0].data.push(data[0].temperature);
        if (mainChart.data.datasets[0].data.length > 40) mainChart.data.datasets[0].data.shi

        mainChart.update();
    }
}
```

Рис. 3.5. Обробка телеметрії та оновлення графіка

Взаємодія клієнтської частини з сервером у реальному часі реалізується за допомогою протоколу WebSocket [11]. У файлі index.html визначено функцію connectWebSocket(), яка встановлює з’єднання з ендпоінтом /ws/telemetry.

Обробка вхідних повідомлень здійснюється у події onmessage, де виконується JSON-парсинг отриманих даних. Залежно від типу повідомлення (telemetry або alert) дані передаються до відповідних функцій обробки (handleTelemetry() та addAlert()).

Особливістю реалізації є наявність механізму автоматичного перепідключення. У події onclose передбачено повторний виклик функції connectWebSocket() через 3 секунди, що забезпечує відновлення зв'язку без участі користувача при тимчасовій втраті з'єднання. Такий підхід гарантує безперервність отримання даних у реальному часі навіть в умовах нестабільної промислової мережі.

Для відображення динаміки технологічних параметрів використано бібліотеку Chart.js. Графік реалізовано у вигляді багатосерійної лінійної діаграми, на якій одночасно відображаються три параметри: температура, тиск води та швидкість прокатки. Використання двох вертикальних осей (ліва – для температури, права – для тиску та швидкості) дозволяє коректно візуалізувати дані з різними діапазонами значень.

Оновлення графіка відбувається асинхронно при отриманні нових даних через WebSocket. У функції handleTelemetry() до відповідних масивів datasets додаються нові значення, після чого викликається метод update(). Для запобігання надмірного зростання обсягу даних реалізовано обмеження на кількість точок (максимум 40 останніх значень), що забезпечує плавну роботу графіка навіть при тривалій експлуатації.

Інтерфейс дашборду розроблено з урахуванням специфіки промислового середовища. Застосовано темну колірну схему, яка знижує навантаження на зір оператора при тривалій роботі в умовах цехового освітлення. Візуальні елементи (KPI-картки, блоки зон, алерти) мають чітку ієрархію та використовують колірне кодування станів: зелений – норма, жовтий – попередження, червоний – критичне відхилення.

Для підвищення зручності роботи реалізовано рольову модель доступу. Залежно від обраної ролі користувача (Диспетчер, Технолог, Адміністратор) динамічно змінюється доступність окремих елементів інтерфейсу. Зокрема, кнопка підтвердження алертів доступна лише для ролі «Диспетчер». Інформація про поточного користувача зберігається в localStorage, що забезпечує збереження сесії між перезавантаженнями сторінки.

Таким чином, клієнтська частина забезпечує зручну, інформативну та ергономічну взаємодію з системою моніторингу, відповідає вимогам промислових дашбордів та інтегрована з серверною частиною через надійний канал WebSocket [11].

3.4 Опис інтерфейсу користувача веб-додатка

Інтерфейс веб-додатка виконано у вигляді сучасного односторінкового застосунку з темною темою, орієнтованого на використання в умовах промислового цеху. Головна панель дашборду забезпечує комплексний моніторинг процесу термомеханічного зміцнення арматури в реальному часі.

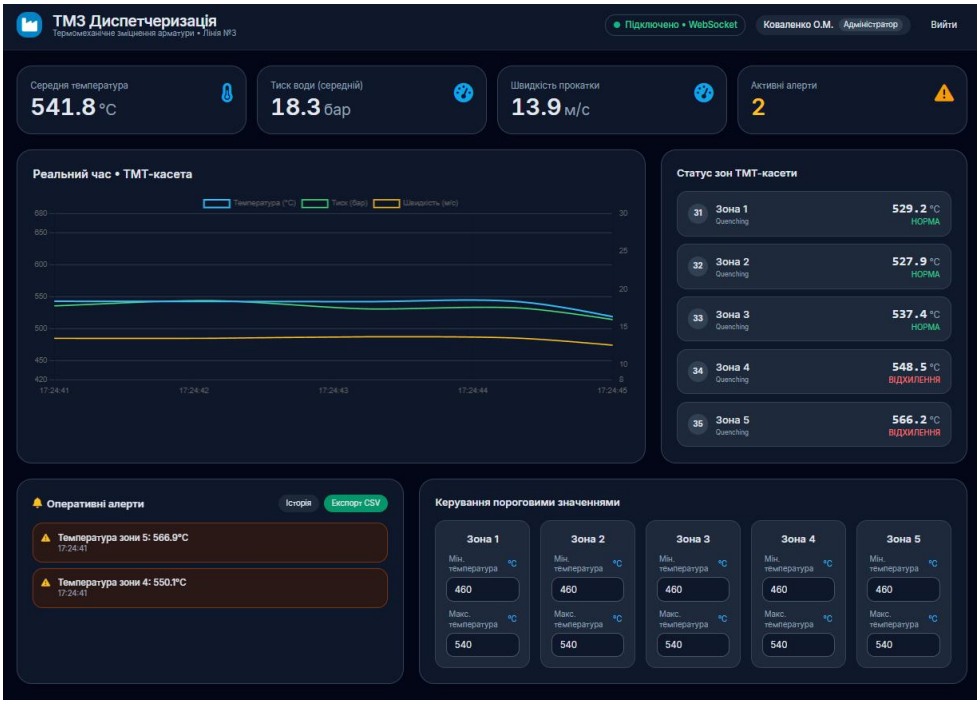


Рис. 3.5. Головна панель дашборду веб-додатка моніторингу та диспетчеризації процесу ТМЗ

Інтерфейс умовно розділено на кілька функціональних блоків: ключові показники ефективності (KPI), графік телеметричних даних, статус технологічних зон та блок оперативних алертів.

Блок KPI відображає середні значення температури, тиску води та швидкості прокатки, а також кількість активних алертів. Графік реального часу побудовано за допомогою бібліотеки Chart.js і дозволяє одночасно спостерігати за динамікою трьох параметрів.

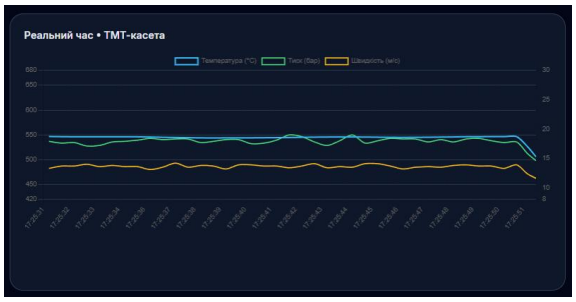


Рис. 3.6. Інтерактивний графік телеметричних даних у реальному часі

Статус кожної з п’яти зон ТМТ-касети візуалізується окремо з кольоровим кодуванням: зелений – норма, червоний – відхилення. Це дозволяє диспетчеру швидко виявляти проблемні зони.

Рис. 3.7. Візуалізація статусу технологічних зон ТМТ-касети

При виході температури за межі встановлених порогових значень система автоматично формує алерти з зазначенням зони та часу виникнення події. Алерти відображаються в хронологічному порядку та мають відповідне колірне виділення.

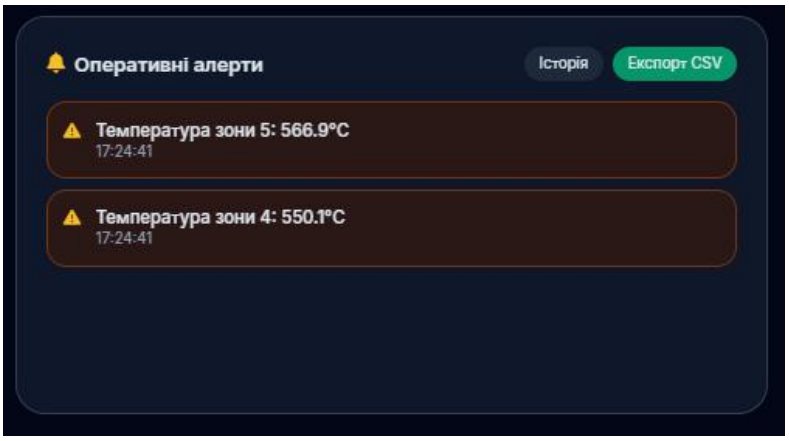


Рис. 3.8. Блок оперативних алертів

Для гнучкого налаштування системи передбачено блок керування пороговими значеннями температури для кожної зони. Зміна порогів одразу впливає на логіку формування алертів.

Таким чином, розроблений інтерфейс забезпечує високу інформативність, зручність сприйняття даних та можливість оперативного реагування на технологічні відхилення в процесі термомеханічного зміцнення арматури.

Висновки до розділу 3

У третьому розділі було розроблено програмне забезпечення веб-додатка для моніторингу та диспетчеризації процесу термомеханічного зміцнення арматури.

На основі сформульованих функціональних та нефункціональних вимог обрано архітектуру та технологічний стек: FastAPI для серверної частини, WebSocket для передачі даних у реальному часі, SQLite для зберігання телеметрії та алертів, а також чисті технології фронтенду (HTML5 + Tailwind CSS + vanilla JavaScript).

Реалізовано емулятор датчиків температури на основі закону Ньютона з додаванням гаусівського шуму та алгоритму ковзного середнього, що дозволяє отримувати реалістичні телеметричні дані для п’яти зон ТМТ-касети. Розроблено логіку автоматичного формування алертів з дедуплікацією та збереженням у базі даних.

Створено клієнтську частину у вигляді односторінкового застосунку з інтерактивним дашбордом, графіками реального часу, системою алертів та рольовою моделлю доступу. Інтерфейс адаптовано під промислові умови експлуатації (темна тема, колірне кодування станів, мінімізація когнітивного навантаження).

Таким чином, у розділі 3 повністю реалізовано поставлену задачу – створено працездатний веб-додаток моніторингу та диспетчеризації, який відповідає сформульованим вимогам та може бути використаний як прототип для впровадження на металургійному підприємстві.

ВИСНОВКИ

У кваліфікаційній роботі було розроблено веб-додаток для моніторингу та диспетчеризації системи термомеханічного зміцнення арматури промислового підприємства.

У результаті виконання роботи було вирішено такі завдання:

– проведено аналіз предметної області та технологічного процесу термомеханічного зміцнення арматури;

– здійснено огляд існуючих аналогів SCADA-систем та IoT-платформ;

– сформульовано функціональні та нефункціональні вимоги до веб-додатка;

– розроблено математичну модель емуляції телеметричних даних;

– спроєктовано архітектуру бази даних та інформаційну модель системи;

– реалізовано серверну та клієнтську частини веб-додатка з використанням технологій FastAPI та WebSocket;

– створено інтерактивний дашборд з графіками реального часу, системою алертів та рольовою моделлю доступу.

Наукова новизна роботи полягає в розробці спеціалізованої архітектури веб-додатка, адаптованої до багатозонної системи термомеханічного зміцнення арматури з урахуванням особливостей реального металургійного виробництва.

Практичне значення отриманих результатів полягає у створенні працездатного прототипу системи моніторингу, який може бути впроваджений на металургійних підприємствах для підвищення якості продукції, зниження браку та оперативності диспетчеризації технологічного процесу.

Розроблений веб-додаток відповідає сучасним вимогам Industry 4.0 та має перспективи подальшого розвитку (інтеграція з MES/ERP-системами, перехід на PostgreSQL, впровадження JWT-автентифікації).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Глушаков А. І. Промислові мережі та протоколи Industry 4.0: монографія. Київ: Політехніка, 2022. 284 с.
2. Гунько Є. М. Моніторинг об’єктів на основі IoT-платформ: навчальний посібник. Кривий Ріг: ДУЕТ, 2023. 198 с.
3. Інформаційні технології в промисловості: збірник наукових праць. Київ: НАУ, 2024. Вип. 15. 312 с.
4. Крижанівський В. І. Сучасні SCADA-системи та їх застосування в промисловості: підручник. Львів: Львівська політехніка, 2021. 356 с.
5. Вакуленко І. О. Термічна обробка металів і сплавів : навчальний посібник. Дніпро: Пороги, 2020. 412 с.
6. ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура та правила оформлювання. Київ: Держстандарт, 2015. 28 с.
7. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ: Держстандарт, 2015. 32 с.
8. FastAPI Documentation [Електронний ресурс]. Режим доступу: https://fastapi.tiangolo.com (дата звернення: 01.06.2026).
9. React – A JavaScript library for building user interfaces [Електронний ресурс]. Режим доступу: https://react.dev (дата звернення: 01.06.2026).
10. PostgreSQL Documentation [Електронний ресурс]. Режим доступу: https://www.postgresql.org/docs (дата звернення: 01.06.2026).
11. WebSocket Protocol [Електронний ресурс] // RFC 6455. 2011. Режим доступу: https://datatracker.ietf.org/doc/html/rfc6455.
12. Industry 4.0: The Future of Productivity and Growth in Manufacturing. Boston Consulting Group, 2023. 48 p.
13. Швайка О. М. Автоматизація технологічних процесів у металургії: підручник. Київ: Знання, 2022. 376 с.

14. Моделювання та оптимізація систем реального часу: навчальний посібник / за ред. С. Г. Глушакова. Кривий Ріг: ДУЕТ, 2024. 214 с.

15. Офіційна документація Recharts [Електронний ресурс]. Режим доступу: https://recharts.org (дата звернення: 01.06.2026).

16. OWASP WebSocket Security Cheat Sheet [Електронний ресурс]. Режим доступу: https://cheatsheetseries.owasp.org/cheatsheets/WebSocket_Security_Cheat_Sheet.html (дата звернення: 01.06.2026).

17. Арсенюк В. Г. Кібербезпека промислових систем управління. Київ: НАУ, 2023. 289 с.

18. Методичні рекомендації щодо виконання та захисту кваліфікаційної роботи бакалавра для здобувачів вищої освіти спеціальності 122 «Комп'ютерні науки» / уклад. С. Глушко, В. Соловйова, Л. Шокотько. Кривий Ріг : ДУЕТ, 2023. 35 с.

19. Вимоги з оформлення письмових робіт у Державному університеті економіки і технологій (нова редакція). Кривий Ріг: ДУЕТ, 2025. 57 с.

20. ISO 6935-2: Steel for the reinforcement of concrete – Part 2: Ribbed bars. Geneva : ISO, 2019.

21. ДСТУ 3760:2019. Арматура періодичного профілю для залізобетонних конструкцій. Технічні умови. Київ: Держстандарт, 2019.

22. Docker Documentation [Електронний ресурс]. Режим доступу: https://docs.docker.com (дата звернення: 01.06.2026).

23. Uvicorn – ASGI server [Електронний ресурс]. Режим доступу: https://www.uvicorn.org (дата звернення: 01.06.2026).

24. TanStack Query Documentation [Електронний ресурс]. Режим доступу: https://tanstack.com/query (дата звернення: 01.06.2026).

25. Вимоги до оформлення дисертацій та авторефератів : наказ МОН України № 40 від 12.01.2017.

Додатки

Додаток А

Структура бази даних SQLite

База даних містить дві таблиці: telemetry_logs для зберігання історичних показників датчиків та alerts для фіксації подій відхилень температурних режимів.

Лістинг А.1 — SQL-скрипт створення таблиць бази даних

```
-- Створення таблиці історичних даних телеметрії
CREATE TABLE IF NOT EXISTS telemetry_logs (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    sensor_id TEXT NOT NULL,           -- Ідентифікатор зони (zone_1 ... zone_5)
    temperature REAL NOT NULL,         -- Температура поверхні прутка, °C
    pressure REAL NOT NULL,            -- Тиск води у форсунках, бар
    speed REAL NOT NULL,               -- Швидкість прокатки, м/с
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Створення таблиці алертів (відхилень)
CREATE TABLE IF NOT EXISTS alerts (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    sensor_id TEXT NOT NULL,
    alert_level TEXT NOT NULL,         -- 'warning' або 'critical'
    description TEXT NOT NULL,         -- Опис події
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
    is_resolved INTEGER DEFAULT 0,     -- 0 - активний, 1 - вирішений
    resolved_at DATETIME,
    resolved_by TEXT                   -- Роль користувача, що підтвердив
);

-- Індекси для прискорення запитів
CREATE INDEX IF NOT EXISTS idx_telemetry_timestamp ON telemetry_logs(timestamp);
CREATE INDEX IF NOT EXISTS idx_alerts_timestamp ON alerts(timestamp);
```

Примітка: База даних SQLite обрана для простоти розгортання та достатньої продуктивності при одному записуючому потоці в умовах промислового підприємства.

Додаток Б

Код класу емулятора датчиків SensorEmulator

```python
import random
import asyncio
from collections import deque
from datetime import datetime

class SensorEmulator:
    def __init__(self, sensor_id: str, initial_temp: float = 980.0,
                 k: float = 0.08, noise_std: float = 1.8):
        self.sensor_id = sensor_id
        self.temp = initial_temp
        self.k = k
        self.noise_std = noise_std
        self.buffer = deque(maxlen=5)
        self.ambient = 25.0

    def step(self) -> dict:
        delta = self.k * (self.ambient - self.temp)
        noise = random.gauss(0, self.noise_std)
        self.temp += delta + noise

        self.buffer.append(self.temp)
        avg_temp = sum(self.buffer) / len(self.buffer)

        pressure = round(18.5 + random.uniform(-0.8, 0.8), 2)
        speed = round(14.2 + random.uniform(-0.3, 0.3), 2)

        return {
            "sensor_id": self.sensor_id,
            "temperature": round(avg_temp, 2),
            "pressure": pressure,
            "speed": speed,
            "timestamp": datetime.now().isoformat()
        }
```

Додаток В

Реалізація серверної частини веб-додатка. Ініціалізація бази даних та збереження телеметрії

```python
async def init_db():
    async with aiosqlite.connect(DB_PATH) as db:
        await db.execute("""
            CREATE TABLE IF NOT EXISTS telemetry_logs (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                sensor_id TEXT NOT NULL,
                temperature REAL NOT NULL,
                pressure REAL NOT NULL,
                speed REAL NOT NULL,
                timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
            )
        """)
        await db.execute("""
            CREATE TABLE IF NOT EXISTS alerts (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                sensor_id TEXT NOT NULL,
                alert_level TEXT NOT NULL,
                description TEXT NOT NULL,
                timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
                is_resolved INTEGER DEFAULT 0,
                resolved_at DATETIME,
                resolved_by TEXT
            )
        """)
        await db.commit()


async def save_telemetry_batch(data_list: list):
    async with aiosqlite.connect(DB_PATH) as db:
        for data in data_list:
            await db.execute("""
                INSERT INTO telemetry_logs (sensor_id, temperature, pressure, speed)
                VALUES (?, ?, ?, ?)
            """, (data["sensor_id"], data["temperature"], data["pressure"], data["speed"]))
        await db.commit()


async def save_alert(sensor_id: str, level: str, description: str):
    async with aiosqlite.connect(DB_PATH) as db:
        await db.execute("""
            INSERT INTO alerts (sensor_id, alert_level, description)
            VALUES (?, ?, ?)
        """, (sensor_id, level, description))
        await db.commit()
```

Додаток Г

Реалізація серверної частини веб-додатка. WebSocket-ендпоінт та обробка підключень

```python
active_connections: list[WebSocket] = []

@app.websocket("/ws/telemetry")
async def websocket_endpoint(websocket: WebSocket):
    await websocket.accept()
    active_connections.append(websocket)
    try:
        while True:
            # Генерація телеметрії від усіх емуляторів
            telemetry_data = []
            for emulator in emulators.values():
                data = emulator.step()
                telemetry_data.append(data)

            # Збереження в базу даних
            await save_telemetry_batch(telemetry_data)

            # Перевірка алертів
            for data in telemetry_data:
                check_and_create_alert(data)

            # Розсилка даних усім підключеним клієнтам
            for connection in active_connections:
                await connection.send_json(telemetry_data)

            await asyncio.sleep(0.85)
    except WebSocketDisconnect:
        active_connections.remove(websocket)
```

Додаток Д

Реалізація клієнтської частини веб-додатка. Підключення та обробка WebSocket

```
let socket;

function connectWebSocket() {
    socket = new WebSocket("ws://localhost:8000/ws/telemetry");

    socket.onopen = function() {
        console.log("WebSocket з'єднання встановлено");
        document.getElementById("connection-status").textContent = "Підключено";
        document.getElementById("connection-status").style.color = "#22c55e";
    };

    socket.onmessage = function(event) {
        const dataArray = JSON.parse(event.data);
        handleTelemetry(dataArray);
    };

    socket.onclose = function() {
        console.log("WebSocket з'єднання закрито. Спроба перепідключення...");
        document.getElementById("connection-status").textContent = "Перепідключення...";
        document.getElementById("connection-status").style.color = "#f59e0b";

        setTimeout(() => {
            connectWebSocket();
        }, 3000);
    };

    socket.onerror = function(error) {
        console.error("Помилка WebSocket:", error);
    };
}
```

Додаток Е

Реалізація клієнтської частини веб-додатка. Обробка телеметрії та оновлення графіка

```
function handleTelemetry(dataArray) {
    dataArray.forEach(data => {
        const zone = data.sensor_id;

        // Оновлення KPI
        updateKPI(data);

        // Оновлення статусу зони
        updateZoneStatus(zone, data.temperature);

        // Додавання даних у графік
        if (chart) {
            const now = new Date();
            chart.data.labels.push(now.toLocaleTimeString());

            // Обмеження кількості точок на графіку
            if (chart.data.labels.length > 40) {
                chart.data.labels.shift();
                chart.data.datasets.forEach(dataset => dataset.data.shift());
            }

            // Додавання значень у відповідні датасети
            chart.data.datasets[0].data.push(data.temperature); // Температура
            chart.data.datasets[1].data.push(data.pressure);    // Тиск
            chart.data.datasets[2].data.push(data.speed);       // Швидкість

            chart.update();
        }
    });
}
```

Додаток Ж

Конфігурація розгортання системи

```
version: '3.8'
services:
  backend:
    build: ./backend
    ports:
      - "8000:8000"
    volumes:
      - ./backend:/app
    restart: unless-stopped
  frontend:
    image: nginx:alpine
    ports:
      - "8080:80"
    volumes:
      - ./frontend:/usr/share/nginx/html:ro
    depends_on:
      - backend
    restart: unless-stopped
```

Інструкція з локального запуску:

1. Клонувати репозиторій або скопіювати файли backend/ та frontend/.

2. Встановити Docker та Docker Compose.

3. Виконати: docker-compose up --build

4. Відкрити у браузері: http://localhost:8080 (інтерфейс) та WebSocket на порту 8000.

5. Для промислового впровадження рекомендується налаштувати HTTPS (wss://) та JWT-автентифікацію WebSocket-з’єднань.

ЗГОДА
здобувача вищої освіти

Державного університету економіки і технологій
про перевірку кваліфікаційної роботи на прояви академічного плагіату та розміщення в Репозитарії Університету

Я, _____**Александров Тихін Анатолійович**,

підтримую політику Державного університету економіки і технологій з академічної доброчесності і відкритого доступу.

Засвідчую, що кваліфікаційна бакалаврська робота

«Розробка веб-додатка для моніторингу та диспетчеризації термостатної системи промислового підприємства»___

виконана самостійно та не містить академічного плагіату. Я не надавав і не одержував недозволену допомогу під час підготовки цієї роботи. Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Державного університету економіки і технологій ознайомлена. Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення норм академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

Також я поінформований, що відповідно до «Положення про Репозитарій (електронну базу даних) Державного університету економіки і технологій» зазначена робота буде розміщена в Електронному архіві Університету (Репозитарії ДУЕТ). З умовами такого розміщення ознайомлена.

15.06.26 Александров Т. А.