Міністерство освіти і науки України

Луцький національний технічний університет

(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій

(повне найменування факультету)

Кафедра комп'ютерної інженерії та безпеки

(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

РОЗРОБКА ОХОРОННОЇ СИСТЕМИ ПРИМІЩЕННЯ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ ШТУЧНОГО ІНТЕЛЕКТУ

DEVELOPMENT OF AN INDOOR SECURITY SYSTEM USING ARTIFICIAL INTELLIGENCE TECHNOLOGIES

спеціальність 123 Комп'ютерна інженерія

(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія

(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІм-21
Биков Сергій Олегович

(підпис)

Керівник:
к.т.н., доцент
Христинець Наталія Анатоліївна

(підпис)

Кваліфікаційну роботу
допущено до захисту
«_____» грудня _____ 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Гринюк Сергій Васильович

(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та безпеки*
Ступінь вищої освіти: *магістр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Т.ТЕРЛЕЦЬКИЙ
«_____« _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Бикову Сергію Олеговичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Розробка охоронної системи приміщення з використанням технологій штучного інтелекту*

Керівник роботи *к.т.н., доцент Христинець Наталія Анатоліївна*

затверджені наказом закладу вищої освіти від «17» червня 2025 року № 290/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 10.12.2025р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області, різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Аналіз проблем та підходів до розробки інтелектуальних охоронних систем
Проєктування архітектури охоронної системи
Програмна реалізація та дослідження функціональних можливостей системи
Висновки

5. Перелік графічного (ілюстративного) матеріалу:

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблем та підходів до розробки інтелектуальних охоронних систем	*Христинець Н. А., доцент*		
Проєктування архітектури охоронної системи	*Христинець Н. А., доцент*		
Розробка та тестування охоронної системи	*Христинець Н. А., доцент*		
Нормоконтроль	*Багнюк Н. В., доцент*		
Гарант ОП	*Гринюк С. В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О. І., ст.викладач*		

7. Дата видачі завдання: 18.06.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми*	До 01.08.2025 р.	
2.	*Аналіз проблем та підходів до розробки інтелектуальних охоронних систем*	До 20.08.2025 р.	
3.	*Проєктування архітектури охоронної системи*	До 25.09.2025 р.	
4.	*Розробка та тестування охоронної системи*	До 20.10.2025 р.	
5.	*Висновки та пропозиції*	До 25.10.2025 р	
6.	*Формування списку використаних джерел*	До 27.10.2025 р.	
7.	*Формування додатків*	До 30.10.2025 р.	
8.	*Оформлення ілюстративного матеріалу*	До 05.11.2025 р.	
9.	*Представлення остаточного варіанту кваліфікаційної роботи керівникові*	До 11.11.2025 р.	
10.	*Нормоконтроль*	До 29.11.2025 р.	
11.	*Інструментальна перевірка на академічний плагіат*	До 02.12.2025 р.	
12.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	До 09.12.2025 р.	

Здобувач вищої освіти _____ Биков С. О.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Христинець Н. А.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Биков С. О. Розробка охоронної системи приміщення з використанням технологій штучного інтелекту. Рукопис.

Кваліфікаційна робота магістра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел, додатків.

У першому розділі роботи подано огляд сучасних технологій та підходів до побудови інтелектуальних охоронних систем, розглянуто тенденції розвитку ринку безпекових рішень, роль штучного інтелекту у підвищенні ефективності систем моніторингу, а також проаналізовано технологічні засади проєктування веб-інтерфейсів та питання модульності й розширюваності програмної архітектури.

У другому розділі описано методи та інструменти, застосовані для розробки системи, зокрема використання мовних моделей штучного інтелекту, специфіку створення клієнтської частини на базі React Native, а також обґрунтовано вибір використаних технологій і засобів

Третій розділ присвячений безпосередньому процесу розробки та тестування охоронної системи, включаючи налаштування інфраструктури, розгортання на VPS-сервері, реалізацію інтерфейсу, організацію взаємодії між клієнтом і сервером, а також розробку модулів аналітики та моніторингу подій.

Ключові слова: веб-інтерфейс, моніторинг, дашборд, REST API, аналітика подій.

ANNOTATION

Bykov S. Development of an Indoor Security System Using Artificial Intelligence Technologies. Manuscript.

Qualifying work of a Master's of EP «Computer Engineering» specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2025.

The thesis consists of an introduction, three chapters, conclusions, a list of references, and appendices.

The first chapter provides an overview of modern technologies and approaches to building intelligent security systems. It examines trends in the security solutions market, the role of artificial intelligence in enhancing the efficiency of monitoring systems, and analyzes the technological principles of web interface design as well as issues of modularity and scalability in software architecture.

The second chapter describes the methods and tools used in the system development, including the use of AI language models, the specifics of creating the client-side using React Native, and the rationale behind the selection of technologies and tools.

The third chapter is dedicated to the actual process of developing and testing the security system, including infrastructure setup, deployment on a VPS server, implementation of the interface, organization of client-server interaction, and development of analytics and event monitoring modules.

Keywords: web interface, monitoring, dashboard, REST API, event analytics.

ЗМІСТ

ВСТУП

Охоронні системи є невід’ємною складовою забезпечення безпеки житлових, комерційних та промислових об’єктів. Вони поєднують апаратні засоби контролю доступу, відеоспостереження, сенсори та програмні модулі для обробки сигналів та подій. Інтеграція технологій штучного інтелекту дозволяє системам не лише фіксувати події, а й проводити їх аналітичну обробку, розпізнавати обличчя, оцінювати поведінкові патерни та прогнозувати потенційні загрози. Завдяки таким можливостям сучасні охоронні системи здатні підвищити ефективність реагування на інциденти та мінімізувати ризики порушень безпеки.

Важливим елементом таких систем є веб-інтерфейс, який забезпечує взаємодію користувача з апаратними та програмними компонентами. Використання компонентної архітектури та технологій на кшталт React або React Native дозволяє створювати інтерактивні інтерфейси з можливістю відображення динамічних даних у реальному часі. Web-інтерфейс дозволяє централізовано управляти пристроями, відслідковувати події, аналізувати показники сенсорів і отримувати аналітичні висновки, що робить систему зручною для користувачів і забезпечує високу продуктивність управління безпекою.

Актуальність теми. Сучасний розвиток охоронних систем характеризується стрімким переходом від традиційних централізованих рішень до інтелектуальних, розподілених та взаємопов’язаних систем, які активно використовують штучний інтелект, машинне навчання, аналітику даних і технології Інтернету речей. Системи нового покоління здатні не лише виявляти порушення, а й прогнозувати потенційні загрози, адаптуватися до змін у середовищі та інтегруватися з іншими цифровими сервісами. Водночас сучасні охоронні системи залишаються вразливими до складних сценаріїв атак, мають обмежену автономність та потребують ефективного контролю цілісності. Інтеграція AI-модулів, багаторівневої обробки даних та веб-інтерфейсів відкриває нові можливості для побудови адаптивних, надійних і масштабованих

охоронних рішень, що робить дослідження відповідних технологій та методів надзвичайно актуальним.

Метою дослідження є розробка та обґрунтування архітектури веб-інтерфейсу інтелектуальної охоронної системи з використанням штучного інтелекту, багаторівневої обробки даних та веб-інтерфейсу для забезпечення високої адаптивності, точності прогнозування загроз та цілісності системи.

Об’єктом дослідження є сучасні інтелектуальні охоронні системи, що використовують штучний інтелект, периферійні сенсори, локальні та хмарні обчислювальні модулі, а також веб-інтерфейси для взаємодії користувача із системою.

Предметом дослідження є методи, технології та архітектурні підходи до забезпечення модульності, розширюваності, цілісності та інтеграції AI-модулів у охоронних системах, а також розробка веб-інтерфейсу для ефективного управління та моніторингу.

Для досягнення мети роботи потрібно вирішити наступні завдання:

– проаналізувати сучасний стан розвитку охоронних систем та порівняти їхні ключові технології, переваги та обмеження;

– дослідити технологічні підходи до розробки веб-інтерфейсів і організації взаємодії з REST API, враховуючи питання модульності, масштабованості та тестування цілісності систем;

– проаналізувати сучасні інструменти та методи розробки програмних компонентів, включно з використанням мовних моделей штучного інтелекту та фреймворку React Native для створення клієнтської частини системи;

– розробити структуру та архітектуру охоронної системи, включаючи проектування інтерфейсу користувача, налаштування серверної частини та організацію взаємодії між клієнтом і сервером;

– провести тестування функціональності, стабільності та безпеки розробленої системи, включаючи модулі аналітики та моніторингу подій.

Апробація результатів. Результати роботи представлені публікацією статті у фаховому збірнику «Комп’ютерно-інтегровані технології: освіта, наука, виробництво» [1]. Матеріали апробації наведено в додатку А.

РОЗДІЛ 1
ОГЛЯД СУЧАСНИХ ТЕХНОЛОГІЙ ТА ПІДХОДІВ ДО РОЗРОБКИ ІНТЕЛЕКТУАЛЬНИХ ОХОРОННИХ СИСТЕМ

1.1 Сучасний стан розвитку охоронних систем

Стан розвитку охоронних систем у всьому світі характеризується переходом від традиційних централізованих рішень до інтелектуальних, розподілених та взаємопов’язаних систем, які активно використовують штучний інтелект, машинне навчання, аналітику даних і технології Інтернету речей. Цей процес супроводжується зростанням рівня автономності технічних засобів безпеки, підвищенням точності виявлення загроз та здатністю систем працювати в режимі прогнозної аналітики, що значно зменшує ймовірність помилкових спрацювань і підвищує ефективність реагування на потенційні інциденти.

Якщо класичні охоронні комплекси раніше включали лише базові компоненти, такі як датчики руху, сигналізацію та відеоспостереження, то сучасні системи нового покоління, такі як Ajax Systems Hub 2 Plus, Hikvision AcuSense, Bosch Smart Security або Google Nest Secure, реалізують концепцію «розумного» аналізу подій і інтеграції з іншими цифровими сервісами.

Вони здатні не лише виявляти спробу проникнення, а й аналізувати поведінкові моделі користувачів, розпізнавати обличчя, відрізняти домашніх тварин від людей, прогнозувати потенційні ризики та самостійно адаптуватися до змін у навколишньому середовищі.

Не дивлячись на численні переваги, сучасні охоронні системи мають і певні недоліки. Проведений аналіз фахових джерел, наукових статей, технічних описів виробників і відкритих публікацій за 2024-2025 роки дозволяє зробити висновок, що жодна з існуючих моделей не забезпечує повної автономності прийняття рішень і комплексної адаптації до непередбачуваних сценаріїв.

Основні параметри оцінювання таких систем, як стабільність роботи, точність розпізнавання, час реакції, рівень хибних спрацювань і можливість

інтеграції з іншими сервісами, демонструють суттєві відмінності між виробниками.

Порівняльну характеристику сучасних систем наведено у таблиці 1.1.

Таблиця 1.1 – Оцінка надійності й стабільності моделей охоронних систем

Переваги	Обмеження та ризики використання
Підвищена точність та гнучкість – завдяки AI, системи можуть адаптуватися до середовища й зменшувати хибні спрацювання	Приватність і законодавчі бар'єри – збір і обробка відео і особистих даних
Модульність і масштабованість – архітектури edge та cloud дають змогу додавати нові модулі без значної реконфігурації	Кібербезпека IoT і API – зовнішні під'єднані сенсори становлять потенційні точки атак
Зниження затримок при локальній обробці та можливість часткового аналізу на edge	Ресурсоємність повноцінних моделей – деякі аналітики потребують великої обчислювальної потужності
Гнучкість для інтеграції AI-модулів та можливість додати компоненти логіки або діалогу	Непослідовність рішень LLM у критичних сценаріях – моделі іноді видають суперечливі рекомендації

Як видно з таблиці, попри значні переваги – підвищену гнучкість, ефективність і можливості масштабування – моделі також стикаються з викликами: захистом приватності, вразливістю IoT-компонентів, а також нестабільністю логіки моделей у реальному контексті.

Оскільки інтелектуальні охоронні системи активно інтегруються з хмарними сервісами, мобільними застосунками та голосовими асистентами, це дає змогу керувати безпекою приміщення віддалено й у режимі реального часу. Наприклад, системи Hikvision AcuSense [2-4] застосовують алгоритми глибокого навчання для відокремлення реальних загроз від хибних тривог, спричинених

тінями, рухом листя або тваринами. Bosch Smart Security інтегрує AI для аналізу відеопотоку та ідентифікації нестандартних звуків або рухів, а комплекс Ajax Systems Hub 2 Plus, який розроблений в Україні, використовує захищені протоколи радіозв'язку та хмарну аналітику для побудови адаптивних сценаріїв реагування. Впродовж останніх років активізувалися дослідження у напрямку використання нейронних мереж для передбачення сценаріїв проникнення на основі часових патернів [5-6] та систем з багатофакторною аналітикою контексту загрози [7]. Це свідчить про перехід від реактивних до проактивних методів кіберзахисту, що дозволяє виявляти потенційні атаки ще на етапі формування поведінкових аномалій у мережевому трафіку і створює підґрунтя для розроблення інтелектуальних систем моніторингу безпеки, подібних до представленої в цій роботі архітектури.

Не дивлячись на численні переваги, сучасні охоронні системи мають і певні недоліки. З проаналізованих публікацій, статей, пабліків та фахових інтернет-ресурсів можна зробити висновок, що більAIсть сучасних охоронних систем орієнтовані переважно на фіксацію подій, а не на їхнє передбачення чи інтелектуальний аналіз.

Додатково варто зазначити, що у багатьох сучасних комплексах безпеки недостатньо розвинутий рівень контекстної аналітики, що обмежує можливості для розпізнавання нетипових шаблонів поведінки. Навіть при наявності датчиків руху, відеоаналітики та мережевих модулів, ці компоненти часто функціонують як автономні елементи, не формуючи єдиного інтелектуального середовища для комплексного аналізу загроз. У результаті система фіксує окремі події, але не здатна встановлювати причинно-наслідкові зв'язки або корелювати різні типи сигналів, що знижує ефективність превентивного реагування.

Дані, які зібрані у таблиці 1.2 доводять, що не зважаючи на високу надійність технічних компонентів і використання зашифрованих каналів зв'язку, такі системи часто залишаються вразливими до складних сценаріїв атак, що базуються на соціальній інженерії або багатокрокових вторгненнях. Це пояснюється обмеженою інтеграцією між підсистемами відеоспостереження,

контролю доступу та мережевого моніторингу, які нерідко функціонують ізольовано, без обміну контекстною інформацією. Основними перевагами є високий рівень адаптивності, мінімізація людського фактора, точність розпізнавання загроз і можливість інтеграції з іншими системами безпеки.

Таблиця 1.2 – Порівняльна характеристика сучасних інтелектуальних охоронних систем (2024-2025 рр.)

Назва системи	Рік	Ключові технології	Переваги	Недоліки
Ajax Systems Hub 2 Plus	2024	IoT, AI-аналітика, захищені протоколи	Висока автономність, мобільний контроль, інтеграція з датчиками	Залежність від хмарних сервісів
Hikvision AcuSense	2024	Deep Learning, Smart Detection	Мінімізація хибних спрацювань, ефективність відеоаналізу	Обмежена сумісність із системами інших брендів
Bosch Smart Security	2025	AI Audio Recognition, Cloud Analytics	Високоточне розпізнавання звуків, масштабованість	Висока ціна, потреба у постійному оновленні
Google Nest Secure	2024	Cloud AI, Home Automation	Інтеграція з екосистемою Google Home, зручний контроль	Конфіденційність даних, залежність від сервера

Серед недоліків виділяють високу вартість установлення та обслуговування, ризики кібератак через підключення до мережі, залежність від стабільного інтернет-з’єднання й потребу у періодичному оновленні програмного забезпечення для підтримки актуальних алгоритмів безпеки

На рисунку 1.1 подано архітектуру сучасної системи відеоаналітики [8], яка демонструє роботу алгоритмів машинного навчання у процесі фільтрації відеопотоку та виділення релевантних подій.

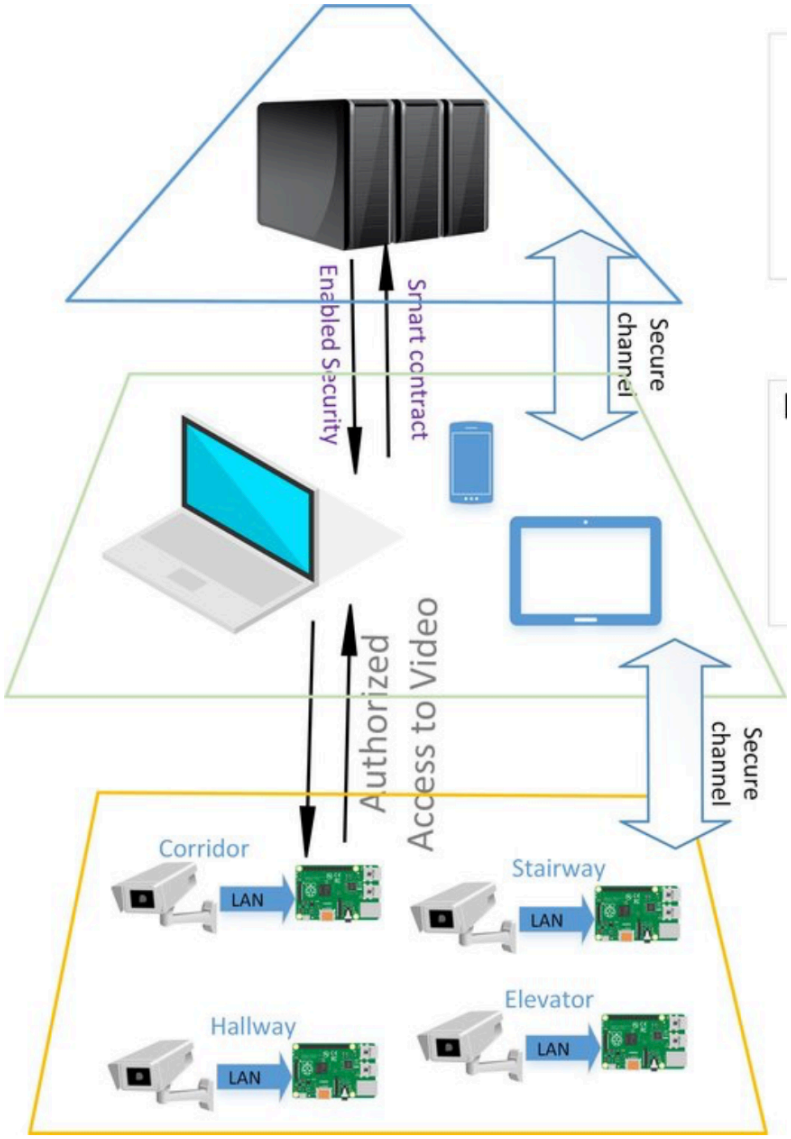

Рисунок 1.1 – Тришарова архітектура edge-fog-cloud для систем відеоспостереження [9]

Зображена архітектура демонструє типову структуру інтелектуальної охоронної системи приміщення, яка поєднує мережеві камери спостереження, обчислювальні вузли, локальні мережі передачі даних, хмарний сервер та користувацькі пристрої доступу – ноутбук, планшет або смартфон. Нижній рівень системи складають периферійні пристрої: відеокамери, розміщені у таких

ключових зонах як коридор, сходова клітка, ліфт, підключені через LAN до локальних обчислювальних модулів, що виконують первинну обробку відеопотоку. Ці модулі можуть використовувати нейромережеві моделі для виявлення аномалій або подій безпеки безпосередньо на периферії, зменшуючи навантаження на центральний сервер.

Середній рівень представлений користувацькими пристроями, які взаємодіють із системою через захищені канали зв'язку. Доступ до відеопотоків здійснюється лише за умови авторизації, що запобігає несанкціонованому перегляду. Тут відбувається управління системою, аналіз отриманих даних, а також взаємодія між користувачами і центральним сервером. З'єднання між рівнями реалізується за допомогою протоколів безпечної передачі даних SSL/TLS, що забезпечують цілісність і конфіденційність відеоінформації.

Верхній рівень представлений сервером або хмарним середовищем, у якому реалізовано механізми керування політиками доступу, зберігання історичних записів та інтелектуальної аналітики. Тут також використовується смарт-контракт – компонент, що автоматизує перевірку прав доступу, автентифікацію користувачів та виконання дій без втручання людини. Така архітектура відповідає принципам розподіленої безпеки, коли рішення приймаються на різних рівнях – від периферійних пристроїв до хмарних сервісів, що дозволяє системі функціонувати навіть у разі локальних збоїв.

Позитивними сторонами цієї архітектури є її масштабованість, гнучкість і високий рівень кіберзахисту. Використання смарт-контрактів створює можливість реалізації автономних політик безпеки, коли система самостійно приймає рішення про надання або обмеження доступу. Інтеграція периферійної обробки відео дозволяє значно зменшити затримку при реагуванні на події, а також оптимізувати використання мережевих і серверних ресурсів. Крім того, архітектура підтримує модульний принцип, що спрощує додавання нових пристроїв, сенсорів або алгоритмів без необхідності повного перезапуску системи.

У кваліфікаційній роботі представлена аналогічна архітектура, яка буде використана для обґрунтування вибору серверної частини веб-інтерфейсу охоронної системи, організації обміну даними між клієнтською частиною та сервером, а також для забезпечення безпечної та стабільної взаємодії користувача з системою в режимі реального часу.

Така архітектура має і низку недоліків, які є потенційними напрямами для подальших досліджень. По-перше, складність інтеграції компонентів різних виробників створює ризик несумісності програмних протоколів і підвищує вимоги до налаштування системи. По-друге, висока залежність від безперебійного функціонування мережевої інфраструктури та хмарних сервісів може призвести до тимчасової втрати доступу у разі збоїв у з'єднанні або перевантаження серверів. По-третє, навіть за наявності шифрування, ризики кібератак залишаються актуальними, особливо у випадку використання слабких алгоритмів автентифікації або неоновлених прошивок периферійних пристроїв.

Зазначені потенційні недоліки були враховані при розробці веб-інтерфейсу охоронної системи.

Для мінімізації ризику несумісності між компонентами різних виробників було обрано стандартизовані протоколи обміну даними та створено централізовану логіку обробки запитів через серверну частину, що забезпечує уніфіковану взаємодію всіх елементів системи. Зокрема, для виконання HTTP-запитів до серверної частини було вирішено використати бібліотеку Axios, застосування якої дозволяє спростити роботу з REST API та надало можливість централізовано налаштовувати заголовки запитів, обробляти помилки та значно зменшити обсяг коду.

Для зменшення впливу можливої нестабільності мережевого з'єднання впроваджено механізми кешування даних на клієнтській стороні та повторну синхронізацію при відновленні доступу, що дозволяє забезпечити безперервність відображення інформації користувачу. Крім того, структура системи передбачає резервування ключових сервісів і оптимізацію використання

серверних ресурсів для підтримки стабільної роботи навіть під високим навантаженням.

На рисунку 1.2 подано узагальнену схему вимог до забезпечення конфіденційності в інтелектуальних охоронних системах.

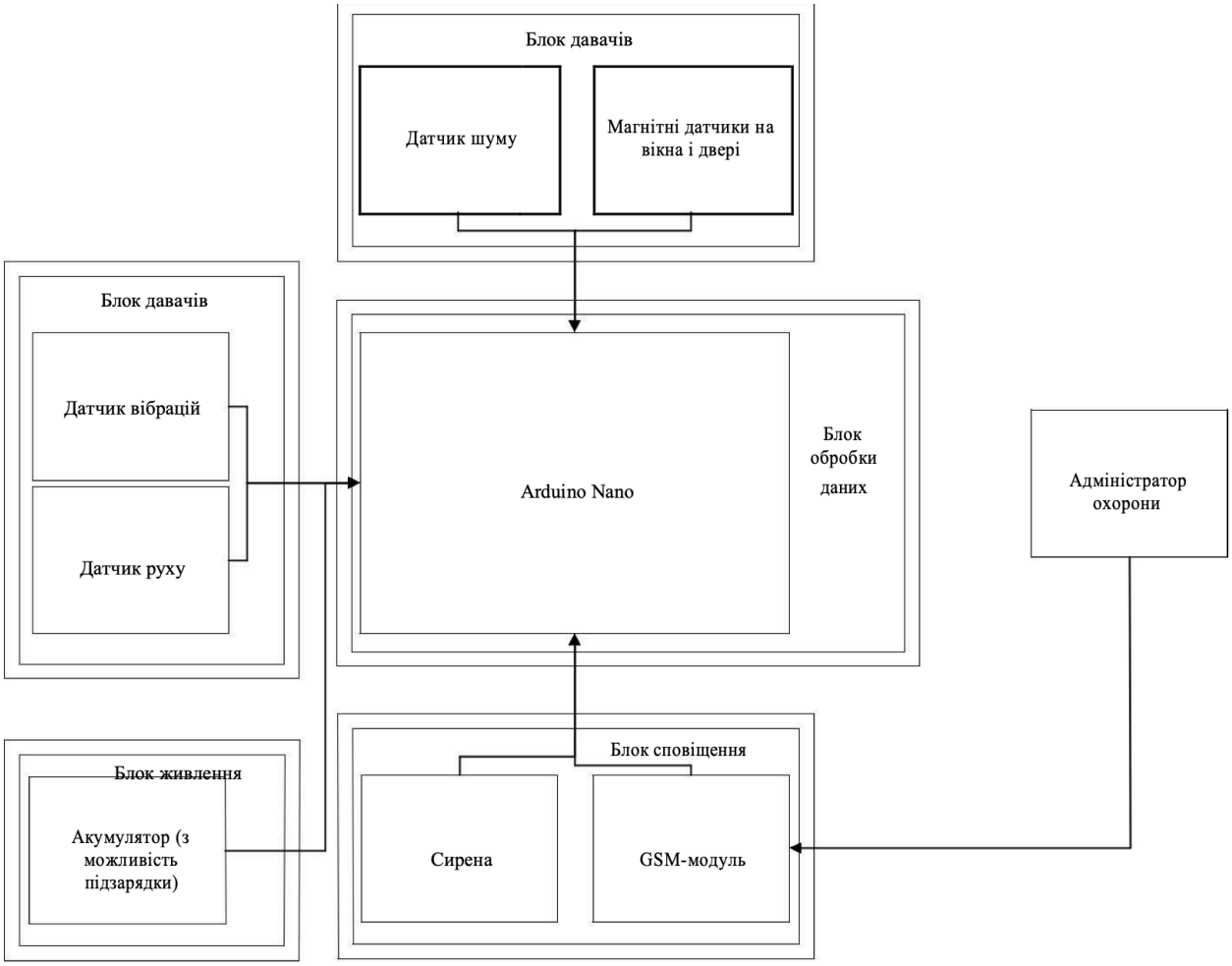

Рисунок 1.2 – Структурна схема охоронної системи [9]

Схема відображає три рівні взаємопов’язаних компонентів: базові вимоги конфіденційності, вимоги, орієнтовані на користувача, та додаткові вимоги до узгодженості даних і якості сервісу.

Базові вимоги конфіденційності формують фундамент системи безпеки та включають такі ключові поняття, як відповідність нормативним вимогам, анонімність користувачів, унеможливлення зв’язування персональних даних із конкретною особою, збереження цілісності інформації та контрольоване

використання зібраних даних. Цей рівень гарантує, що система не лише виявляє загрози, але й захищає приватність користувачів від стороннього доступу, зокрема у процесі передачі даних між сенсорами, сервером і користувацьким інтерфейсом.

Другий рівень на схемі визначає вимоги, що орієнтовані на користувача і акцентує увагу на прозорості й контролі з боку власника даних. Сюди належать отримання згоди користувача на обробку інформації, право на володіння власними даними, релевантність зібраних відомостей до мети обробки та захист персональної ідентифікаційної інформації. Ці елементи відіграють важливу роль у сучасних охоронних рішеннях, які дедалі частіше інтегрують інтелектуальні аналітичні сервіси, здатні розпізнавати обличчя, аналізувати поведінкові патерни або використовувати дані біометрії.

Третій рівень – це додаткові вимоги до конфіденційності охоплює узгодженість інформації між компонентами системи та забезпечення належного рівня якості обслуговування QoS. Для інтелектуальних охоронних систем це означає синхронізовану роботу камер, датчиків і серверів у реальному часі без втрати даних, а також стабільне функціонування мережевої інфраструктури під навантаженням.

Розглянута схема була використана в рамках кваліфікаційної роботи як орієнтир для побудови структури веб-інтерфейсу охоронної системи та визначення ключових вимог до захисту даних і узгодженості інформаційних потоків. Вона забезпечила основу для організації обміну даними між клієнтською частиною, реалізованою на JavaScript і React, та серверною частиною на PHP, дозволяючи гарантувати безпечну, масштабовану та стабільну взаємодію між компонентами.

Завдяки інтеграції принципів конфіденційності, контролю користувача та забезпечення QoS у проекті було досягнуто синхронізованої роботи системи, ефективного управління динамічними даними та високої надійності функціонування веб-інтерфейсу.

1.2 Використання штучного інтелекту в системах безпеки

Історично методи управління технічним обслуговуванням ліфтового обладнання пройшли значну еволюцію, що відображає загальні тенденції розвитку інформаційних технологій у промисловості. Цю еволюцію можна умовно поділити на декілька етапів, кожен з яких характеризується певним рівнем автоматизації та типом використовуваних інструментів.

У сучасних охоронних системах застосування штучного інтелекту стає ключовим чинником, який відкриває новий рівень функціональності – не лише фіксування подій, але й інтелектуальний аналіз і прогнозування. Одним із важливих напрямів є розпізнавання образів та облич, яке дозволяє автоматично ідентифікувати особу за відеопотоком. У цьому процесі згорткові нейронні мережі здатні вилучати ознаки, що відрізняють одну особу від іншої, і співставляти їх із існуючими базами даних згідно формули (1.1). Формально ця модель класифікації описана в огляді [10]:

$$\hat{y} = \arg\max_{y \in Y} P(y \mid X, \theta), \tag{1.1}$$

де X – вхідне зображення;

Y – множина класів-ідентифікаторів;

θ – параметри моделі;

$P(y \mid X, \theta)$ – ймовірність належності зображення до класу.

Окрім розпізнавання, важливе значення набуває аналіз поведінкових патернів за допомогою рекурентних або гібридних нейронних мереж, які здатні моделювати часові залежності та виділяти аномальні дії. У таких системах функція втрат середньоквадратичної похибки (2) використовується для оцінки відповідності фактичних параметрів (x_i) та прогнозованих ($\hat{x}_i$), що обраховують за формулою (1.2).

$$L = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{x}_i)^2 \quad (1.2)$$

де i – число використаних рекурсій ;

N – рівні моделі.

Якщо значення L перевищує адаптивно встановлений поріг δ, система класифікує подію як потенційну загрозу. Такий підхід детально розглядається в дослідженнях з виявлення аномалій у мережевому трафіку та поведінкових сигналах [11-12].

Штучний інтелект також активно використовується для оптимізації ресурсів системи охорони – зокрема енергії, пам'яті та мережевого трафіку. Система може формулювати задачу (3) мінімізації суми енергоспоживання та затримки за формулою (1.3).

$$\min_{r_i} \quad \sum_{i=1}^{n} (E_i(r_i) + T_i(r_i)), \quad (1.3)$$

де $E_i(r_i)$ – енергоспоживання вузла i при ресурсах r_i;

$T_i(r_i)$ – часові затрати на виконання завдань;

n – кількість вузлів.

У науковій літературі цей підхід досліджується в контексті енергоефективних архітектур edge-cloud [13-14].

Ще одним перспективним напрямом є прогнозування інцидентів на базі історичних даних. Тут використовуються моделі довготривалої пам'яті, так звані Long Short-Term Memory, які вміють обробляти часові послідовності. Ймовірність виникнення інциденту описується формулою (4) у майбутньому моменті $t + k$ може бути виражена формулою (1.4).

$$P(I_{t+k} = 1 \mid X_t, X_{t-1}, \dots, X_{t-n}) = f_{\text{LSTM}}(X), \quad (1.4)$$

де X_t – вектор вхідних параметрів у момент t;

f_{LSTM} – функція передбачення.

Застосування цих моделей в охороні активно висвітлюється у сучасних публікаціях [15-16].

Інтеграція моделей штучного інтелекту у локальні або хмарні середовища дозволяє формувати адаптивні механізми реакції, коли система самостійно обирає сценарій дій залежно від контексту події. Для цього між модулями потрібен стандартизований обмін даними, використовуючи протоколи REST API, MQTT або WebSocket, що забезпечує узгодженість і масштабованість системи [17-18].

Українські науковці також досліджують питання інтелектуального аналізу даних у контексті складних систем. Наприклад, автори навчального посібнику «Інтелектуальний аналіз даних» розглядають методи обробки великих даних і застосування нейронних мереж у прикладних завданнях, що може бути адаптовано до охоронних систем [19].

У підручнику «Основи теорії та проєктування систем охоронної та пожежної сигналізації» висвітлено базові принципи побудови систем безпеки й може слугувати методичною основою для побудови гібридних рішенням із компонентами штучного інтелекту [20].

Таким чином, застосування штучного інтелекту у сфері охоронних систем дозволяє здійснити перехід від реактивного реагування до проактивного прогнозування. Використання моделей розпізнавання, аналізу поведінки, оптимізації ресурсів і прогнозування у поєднанні зі стандартизованими протоколами комунікації формує архітектуру інтелектуальних систем, здатних адаптуватися до нових загроз у режимі реального часу.

1.3 Аналіз технологічних основ побудови веб-інтерфейсів у взаємодії з REST API

Веб-інтерфейси вбудованих систем, в тому числі і охоронних, виконують роль центрального інструменту взаємодії користувача з системою: моніторинг

датчиків, управління пристроями, перегляд подій та сповіщень у реальному часі. REST API забезпечує стандартизовану, легку та масштабовану комунікацію між фронтендом і сервером.

Фактично, REST API визначає набір HTTP-запитів у вигляді GET, POST, PUT, DELETE, що дозволяють клієнтським веб-додаткам отримувати або модифікувати дані з серверного боку.

Є кілька архітектурних підходів до побудови інтерфейсів систем за вказаною комунікацією.

Класична REST-комунікація передбачає те, що фронтенд робить періодичні запити до серверу для оновлення стану датчиків. Комбінована схема REST + WebSocket використана в кваліфікаційній роботі і полягає в тому, що REST використовується для базових операцій, а WebSocket – для отримання подій у реальному часі. Є ще мікросервісний підхід, коли бекенд розбивається на сервіси – датчики, сповіщення, користувачі – кожен з яких надає REST API, а фронтенд агрегує дані.

На рисунку 1.3 зображено архітектурний підхід [21], що ґрунтується на інтеграції REST та WebSocket технологій для забезпечення двонапрямленої взаємодії між клієнтськими застосунками та серверною частиною системи. Така архітектура поєднує принципи клієнт-серверної моделі з елементами подієво-орієнтованої комунікації, що забезпечує як синхронну, так і асинхронну передачу даних.

Основою функціонування є веб-сервер, який обробляє стандартні HTTP-запити від клієнтів через REST API, забезпечуючи запит-відповідь взаємодію.

Паралельно реалізується канал постійного з’єднання через WebSocket Gateway, який підтримує безперервний зв’язок між клієнтськими вебзастосунками та сервером у режимі реального часу.

Це дозволяє не лише надсилати дані на запит, а й здійснювати зворотну передачу інформації у відповідь на події без повторних запитів з боку клієнта.

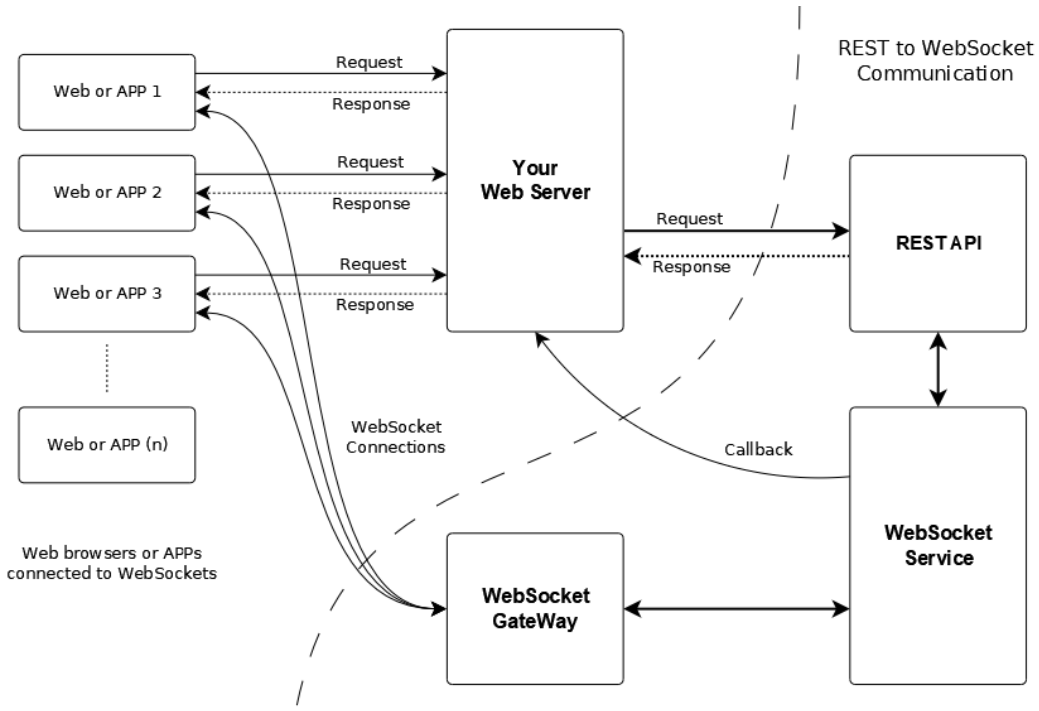

Рисунок 1.3 – Гібридна архітектура взаємодії REST та WebSocket [21]

WebSocket Service взаємодіє з REST API, забезпечуючи внутрішню комунікацію між асинхронним сервісом реального часу та традиційними HTTP-запитами. Таким чином, реалізується гібридна архітектура, у якій REST відповідає за транзакційні, структуровані операції, тоді як WebSocket – за потокову взаємодію та миттєве оновлення даних. Завдяки такому підходу досягається підвищена ефективність обробки подій, зниження затримок при передачі інформації та адаптивність інтерфейсу до динамічних змін стану системи.

1.4 Питання модульності, розширюваності та тестування цілісності системи

Для забезпечення ефективності та надійності охоронних систем ключовими стають принципи модульності, розширюваності та тестування їх цілісності. Ці характеристики визначають можливість системи адаптуватися до змін зовнішнього середовища, масштабуватися та гарантувати коректність взаємодії між її компонентами.

Модульність охоронної системи передбачає поділ загальної функціональності на незалежні, логічно завершені компоненти, які реалізують окремі функції без порушення роботи всієї системи. Такий підхід дозволяє проектувати гнучку структуру, що полегшує розробку, налагодження та модернізацію. Кожен модуль може виконувати свою роль – наприклад, збір даних від сенсорів, аналіз подій, управління доступом, взаємодію з користувачем або обробку сигналів тривоги.

З технічної точки зору, модульна архітектура дозволяє використовувати принципи Service-Oriented Architecture або Microservices Architecture, що забезпечує слабке зв’язування компонентів через стандартизовані інтерфейси. Це дає можливість інтегрувати нові функціональні блоки без суттєвих змін у загальній системі, зберігаючи при цьому узгодженість протоколів обміну даними. Модульність також спрощує оновлення окремих частин системи без потреби її повної реконфігурації, що є важливим у контексті довготривалої експлуатації охоронних комплексів та постійного розвитку технологій.

Розширюваність охоронної системи означає можливість додавання нових функціональних можливостей або підсистем без зниження продуктивності та стабільності роботи. Архітектурно це досягається за рахунок гнучкої системи інтеграційних інтерфейсів і підтримки протоколів, які дозволяють підключати додаткові сенсори, аналітичні модулі чи програмні сервіси.

У сучасних системах безпеки часто використовується підхід горизонтальної масштабованості, за якого збільшення кількості пристроїв або користувачів компенсується додаванням нових серверів чи вузлів обробки даних. Такий підхід особливо ефективний для систем, що використовують хмарну або гібридну інфраструктуру, де розподілена обробка подій і централізований моніторинг забезпечують високий рівень надійності та доступності.

Завдяки розширюваності система може бути адаптована до конкретних умов експлуатації: від невеликих локальних об’єктів до розподілених інтелектуальних мереж, що об’єднують безліч пристроїв Інтернету речей.

Важливим моментом є також підтримка зворотної сумісності, що дозволяє інтегрувати нові компоненти із вже існуючими підсистемами без повної заміни технічного забезпечення.

Цілісність охоронної системи визначається її здатністю функціонувати відповідно до заданих специфікацій, забезпечуючи безперервність логічних і фізичних зв’язків між модулями. Тестування цілісності охоплює перевірку як програмних, так і апаратних складових.

На програмному рівні здійснюється верифікація коректності обміну даними між модулями, тестування API, а також перевірка узгодженості алгоритмів обробки подій. Застосовуються такі методи, як юніт-тестування, інтеграційне тестування, а також моделювання критичних сценаріїв.

На апаратному рівні перевіряється фізична надійність з’єднань, працездатність сенсорів, коректність передачі сигналів тривоги, а також функціонування каналів резервного живлення. Для цього використовують методи автоматизованого моніторингу, що дозволяють фіксувати порушення цілісності в режимі реального часу. Ключовим підходом є використання системи контролю цілісності, яка виконує періодичний аудит внутрішніх компонентів, перевіряє відповідність контрольних сум, конфігураційних файлів та поточних даних очікуваним параметрам. Такий підхід мінімізує ризики саботажу або несанкціонованих змін у конфігурації системи.

Ці три характеристики перебувають у тісному взаємозв’язку. Висока модульність забезпечує можливість розширення системи без порушення її цілісності, тоді як систематичне тестування цілісності гарантує стабільність роботи після додавання нових компонентів. У результаті формується адаптивна архітектура, здатна еволюціонувати без втрати функціональної надійності.

На відміну від традиційних систем, інтелектуальні охоронні комплекси характеризуються складною багаторівневою структурою, інтеграцією різнорідних сенсорів, аналітичних модулів, мережевих компонентів та елементів штучного інтелекту. У такому середовищі навіть незначні порушення

узгодженості або втрати даних можуть призвести до деградації точності виявлення загроз, некоректної реакції системи або втрати керованості.

Тестування цілісності інтелектуальних охоронних систем розглядаються у багатьох наукових джерелах [22-25] як багаторівневу задачу, що охоплює апаратний, програмний і інформаційний рівні.

На рисунку 1.4 представлена схема цілісності інтелектуальних систем.

Рисунок 1.4 – Багаторівнева модель тестування цілісності інтелектуальних охоронних систем

Схема є систематизацією даних із цих джерел і відображає комплексний підхід до перевірки апаратного, програмного та інформаційного рівнів інтелектуальних охоронних систем. На апаратному рівні основна увага приділяється перевірці довіреної початкової завантажувальної ланки та механізмів secure boot, а також контролю фізичної конфігурації сенсорів і камер.

Дослідження з виявлення фізичних атак на камери демонструють, що методи комп’ютерного зору і глибинного навчання можуть виявляти спроби приховування або підміни кадрів за ознаками спектральних/просторових змін і артефактів, тим самим забезпечуючи тестову процедуру для фізичної цілісності пристрою.

На рівні вбудованого програмного забезпечення та пристроїв IoT ключовою практикою є віддалена атестація виконання операцій і даних. Механізми віддаленої атестації, які формалізують поняття Operation Execution Integrity, дозволяють верифікувати, що конкретна операція на пристрої виконана без несанкціонованих змін контролю потоку або критичних даних.

Запропонована багаторівнева структура дозволяє своєчасно виявляти як фізичні, так і логічні загрози, забезпечуючи стійкість системи до підміни, втрати або спотворення інформації.

РОЗДІЛ 2
МЕТОДИ ТА ІНСТРУМЕНТИ ДОСЛІДЖЕННЯ

2.1 OpenAI GPT-4.5 та Google Gemini

У інтелектуальних охоронних системах дедалі активніше застосовуються засоби штучного інтелекту для аналізу даних, прогнозування подій та оптимізації роботи компонентів. Серед сучасних високопродуктивних мовних моделей особливу увагу привертають OpenAI GPT-4.5 та Google Gemini, які здатні забезпечувати ефективну обробку природної мови, генерацію контенту та підтримку автоматизованих аналітичних процесів у реальному часі.

OpenAI GPT4.5 представляє собою трансформерну модель останнього покоління, що характеризується високою здатністю до контекстного розуміння та адаптивного реагування на запити користувачів. Модель підтримує багатокроковий аналіз даних, інтеграцію знань з різних доменів та генерацію інтелектуальних рекомендацій.

GPT-4.5 у рамках охоронних систем може застосовуватися для аналізу подій і виявлення аномалій у даних журналів, автоматизованого формування сценаріїв реагування на підозрілі дії користувачів та пристроїв, генерації описів подій і звітів у реальному часі, а також для забезпечення інтелектуальних інтерфейсів взаємодії користувача із системою. Особливістю GPT-4.5 є можливість роботи з великими обсягами контексту, що дозволяє коректно інтегрувати історію подій та дані з різних сенсорів, забезпечуючи більш точний прогноз поведінки об’єкта охорони. Крім того, модель підтримує адаптивне налаштування під специфіку конкретної організації чи об’єкта, що підвищує ефективність системи при різних сценаріях загроз.

Google Gemini є інтегрованою мультимодальною моделлю, здатною опрацьовувати текстову, візуальну та сенсорну інформацію. У рамках інтелектуальних охоронних систем вона забезпечує аналіз відео- та аудіопотоків із камер і сенсорів для виявлення потенційних загроз, синтезування структурованих повідомлень про події на основі мультимодальних даних,

підтримку систем прогнозування шляхом інтеграції історичних і поточних параметрів, а також виконання складних сценаріїв автоматизованої реакції з урахуванням інформації від IoT-пристроїв і локальних обчислювальних модулів.

Порівняльний аналіз GPT-4.5 та Google Gemini дозволяє відзначити їх ключові особливості (таблиця 2.1).

Таблиця 2.1 – Основні функції та можливості OpenAI GPT-4.5 і Google Gemini

Параметр	OpenAI GPT-4.5	Google Gemini
Тип моделі	Трансформер, NLP	Мультимодальна: текст, зображення та сенсорні дані
Основна функція	Генерація тексту, аналіз подій	Синтез та аналіз мультимодальної інформації
Переваги	Висока точність у прогнозуванні текстових сценаріїв; інтеграція історичних даних	Комплексний аналіз подій із декількох джерел; можливість розпізнавання аномалій на основі зображень та сенсорних сигналів
Обмеження	Обмежена робота з візуальними даними	Потребує значних обчислювальних ресурсів; складніша інтеграція з існуючими системами

Таким чином, застосування GPT-4.5 та Google Gemini у рамках інтелектуальних охоронних систем забезпечує високий рівень адаптивності та можливості автоматизованого аналізу даних. Використання цих моделей дозволяє здійснювати як проактивний контроль подій, так і створювати адаптивні

сценарії реагування, інтегровані із багаторівневою архітектурою системи, описаною у попередньому розділі 1.4 кваліфікаційної роботи.

2.2 Розробка веб інтерфейсу на React Native

Для реалізації веб-інтерфейсу інтелектуальної охоронної системи у даній роботі використано технологію React Native, яка дозволяє створювати кросплатформні застосунки з єдиним кодовим базисом для мобільних і веб-платформ [26-27]. Використання React Native забезпечує високу інтерактивність інтерфейсу та можливість реактивного оновлення компонентів у реальному часі, що особливо актуально для відображення потокових даних від сенсорів і камер системи безпеки.

Архітектурно веб-інтерфейс побудовано за принципами компонентного підходу, що дозволяє поділити інтерфейс на логічні модулі з визначеною функціональністю, такі як відображення даних сенсорів, журнал подій, панель управління та сповіщення. Кожен компонент інтегрується з серверною частиною через стандартизовані API, що забезпечує узгодженість і масштабованість системи.

Взаємодія з серверною частиною здійснюється через REST API та WebSocket протоколи. REST API відповідає за обмін структурованими даними, зокрема за отримання стану датчиків, оновлення конфігурацій та зберігання історії подій. WebSocket забезпечує двонаправлену передачу даних у режимі реального часу, що дозволяє миттєво відображати зміни стану системи та оперативно реагувати на події. Поєднання цих протоколів дозволяє оптимізувати обсяг мережевого трафіку і забезпечити адаптивність інтерфейсу до динамічних змін.

Розробка веб-інтерфейсу передбачає також інтеграцію механізмів автентифікації і авторизації користувачів, що забезпечує контроль доступу та захист конфіденційної інформації. Використання React Native дозволяє реалізувати сучасні методи управління станом компонентів та обробки подій, що

сприяє стабільності роботи інтерфейсу навіть за високого навантаження або при великій кількості підключених користувачів.

Інструментально процес розробки включав середовище Visual Studio Code, систему контролю версій Git та бібліотеки для побудови інтерфейсних компонентів і роботи з мережевими запитами.

Такий підхід забезпечує гнучкість і масштабованість веб-інтерфейсу, дозволяє швидко додавати нові функціональні блоки та інтегрувати алгоритми штучного інтелекту, забезпечуючи адаптивну взаємодію користувача з охоронною системою.

Розроблений веб-інтерфейс проходить комплексне тестування для забезпечення стабільності, коректності відображення даних і ефективної взаємодії з серверною частиною.

Тестування включає перевірку функціональної відповідності компонентів, обробку подій у режимі реального часу, коректність обміну даними через REST API та WebSocket, а також стрес-тести на навантаження для оцінки продуктивності при великій кількості одночасних користувачів.

Валідація системи охоплює перевірку узгодженості відображених даних із фактичним станом сенсорів і камер, адекватність сповіщень про події та точність роботи алгоритмів автоматизованого реагування.

Особлива увага приділяється тестуванню механізмів автентифікації та авторизації користувачів, що гарантує безпечний доступ до конфіденційної інформації.

Систематичне тестування і валідація дозволяють забезпечити надійну роботу веб-інтерфейсу в умовах реального використання, підтверджують правильність інтеграції з локальними та хмарними обчислювальними модулями і створюють основу для подальшої масштабованої експлуатації системи.

У результаті розроблений інтерфейс забезпечує ефективну взаємодію користувача з охоронною системою, гнучкість управління та адаптивне відображення даних у режимі реального часу.

2.3 Використані технології для розробки веб-інтерфейсу та обґрунтування вибору

Реалізація веб-інтерфейсу охоронної системи вимагала вибору технологічного стеку, здатного забезпечити високу продуктивність, масштабованість та можливість подальшої підтримки розробленого програмного забезпечення. У межах клієнтської частини застосунку використано мову програмування JavaScript у поєднанні з бібліотекою React, що дало змогу сформувати модульну структуру інтерфейсу та забезпечити динамічне оновлення даних у режимі реального часу.

Серверну частину побудовано на основі PHP, який продемонстрував стабільність, адаптивність до роботи з REST API та ефективність у взаємодії з базами даних, механізмами автентифікації й логікою обробки запитів.

Загалом, архітектурна схема технологічного стеку веб-інтерфейсу наведена на рисунку 2.1.

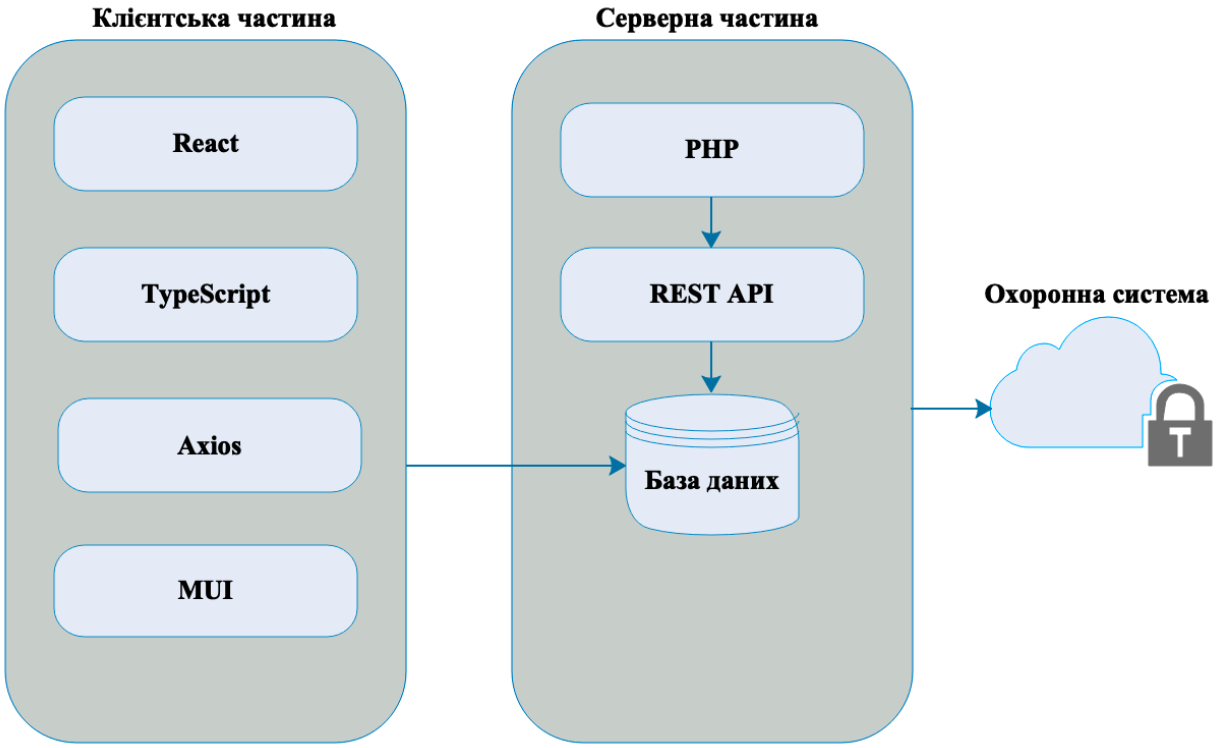

Рисунок 2.1 – Структурна схема взаємодії клієнтської частини, серверної частини та охоронної системи

Серверна частина, реалізована на PHP, надала можливість ефективно обробляти запити, що надходять від клієнтського застосунку, забезпечуючи коректний обмін даними між елементами охоронної системи. Технологія продемонструвала сумісність із REST-архітектурою та відповідність вимогам до стабільності й безперервності функціонування. Зведена таблиця 2.2 демонструє загальні характеристики вибраних технологій:

Таблиця 2.2 – Обґрунтування вибору технологій для розробки веб-інтерфейсу охоронної системи

Використані технології	Призначення	Переваги	Чому обрано для системи
React	Побудова інтерфейсу користувача	Компонентність, швидкість, Virtual DOM	Оптимальний для динамічного інтерфейсу
TypeScript	Статична типізація JS-коду	Менше помилок, краща підтримка	Підвищення надійності фронтенду
Axios	HTTP-запити	Простота, підтримка перехоплювачів	Зручна робота з REST API
MUI	UI-компоненти	Готові елементи, сучасний стиль	Швидке створення інтерфейсу
PHP	Серверна логіка	Стабільність, сумісність	Надійна реалізація REST API

Для оптимізації процесу збірки застосунку використано інструмент Vite.js, який зменшив час ініціалізації проекту порівняно зі стандартними збірниками та спростив інтеграцію з React. Формування єдиного стилістичного середовища інтерфейсу здійснювалося за допомогою бібліотеки MUI, що надала набір

уніфікованих компонентів та дозволила зосередити увагу на реалізації логічної частини системи.

Застосування JavaScript визначено його ключовою роллю в сучасній веб-розробці та здатністю забезпечувати інтерактивність на рівні клієнта без потреби перезавантаження сторінки. Використання бібліотеки React сприяло розподілу інтерфейсу на незалежні компоненти, що полегшило тестування, розширення функціональності й підтримку проекту. Механізм віртуального DOM забезпечив оптимізацію процесів оновлення інтерфейсу, що є критично важливим у системах, орієнтованих на роботу з великими обсягами динамічних даних, зокрема журналами подій, показниками сенсорів та інформаційно-аналітичними модулями.

Взаємодію з REST API організовано засобами бібліотеки Axios, яка забезпечила централізовану конфігурацію запитів, обробку типових помилок та зменшення дублювання коду. Навігацію між сторінками веб-інтерфейсу реалізовано за допомогою react-router-dom, що дало змогу сформувати логічну й безперервну структуру переходів між авторизаційним модулем, панеллю керування, сторінкою відображення сенсорів та системою аналітики.

Додаткове використання TypeScript дозволило підвищити рівень формальної чіткості та передбачуваності коду завдяки статичній типізації, що зменшило кількість можливих помилок під час розробки та покращило структурну цілісність проекту. Для візуалізації статистичних даних застосовано бібліотеку Recharts, за допомогою якої реалізовано інтерактивні графіки й діаграми зі здатністю оновлювати дані у режимі реального часу.

Розробка велась у двох інтегрованих середовищах. Visual Studio Code було використано для клієнтської частини завдяки широкому набору інструментів, підтримці Git та розширених можливостей налагодження. PhpStorm забезпечив повноцінну роботу з серверним середовищем, включаючи автоматичну перевірку синтаксису, інтеграцію з базами даних та контроль структур REST API. Поділ середовищ розробки дозволив підвищити організованість роботи, оптимізувати робочі процеси та забезпечити високу якість програмного коду.

2.4 Інструменти дослідження

Для реалізації поставлених завдань у роботі були використані сучасні програмні та апаратні інструменти, що забезпечують ефективну розробку, тестування та аналіз функціонування охоронної системи.

Мовні моделі штучного інтелекту, зокрема OpenAI GPT-4.5 та Google Gemini, застосовувалися для автоматизації генерації текстових описів подій, аналізу логів та підтримки модулів аналітики. Це дозволило підвищити точність прогнозування поведінкових патернів користувачів і пристроїв та оптимізувати процес обробки інформації.

Фреймворк React Native був використаний для створення кросплатформного веб-клієнта, що забезпечує інтерактивний інтерфейс і підтримку мобільних пристроїв. Застосування TypeScript сприяло типізації даних, підвищенню надійності коду та уникненню помилок при взаємодії компонентів, а бібліотека React забезпечила компонентну архітектуру і оптимізацію оновлення інтерфейсу у реальному часі.

Серверна частина системи була реалізована на PHP, що дозволило ефективно обробляти HTTP-запити, організовувати взаємодію з REST API і забезпечувати обмін даними між усіма елементами охоронної системи. Для потокової взаємодії та миттєвого оновлення даних застосовувався WebSocket, що підвищувало швидкість обробки інформації і дозволяло інтегрувати аналітичні модулі.

Середовище розгортання включало віртуальний приватний сервер з Linux OS та панель керування FastPanel, що забезпечувало стабільну роботу системи у реальному середовищі. Налаштування DNS забезпечувало коректну маршрутизацію запитів від клієнтських пристроїв до сервера і стабільність функціонування веб-застосунку.

Інструменти аналітики та моніторингу включали вбудовані модулі збору та обробки даних про пристрої, події безпеки та показники сенсорів. Сервісні модулі взаємодії з API забезпечували централізовану обробку запитів і передачу

результатів до компонентів клієнтського інтерфейсу, а логування та обробка JSON-даних гарантували узгодженість телеметричних даних, їх точність і зручність для подальшого аналізу та візуалізації.

Завдяки комбінації зазначених інструментів було забезпечено комплексний підхід до розробки, тестування та аналізу охоронної системи, що дозволило досягти високого рівня інтерактивності, стабільності та аналітичних можливостей системи.

РОЗДІЛ 3
РОЗРОБКА ТА ТЕСТУВАННЯ ОХОРОННОЇ СИСТЕМИ

3.1 Налаштування структури проекту та розміщення файлів

Під час розробки веб-інтерфейсу охоронної системи було сформовано впорядковану структуру проєкту, яка забезпечує зручність у підтримці, розширенні та масштабуванні системи. Структура клієнтського застосунку з React-компонентами наведена на рисунку 3.1.

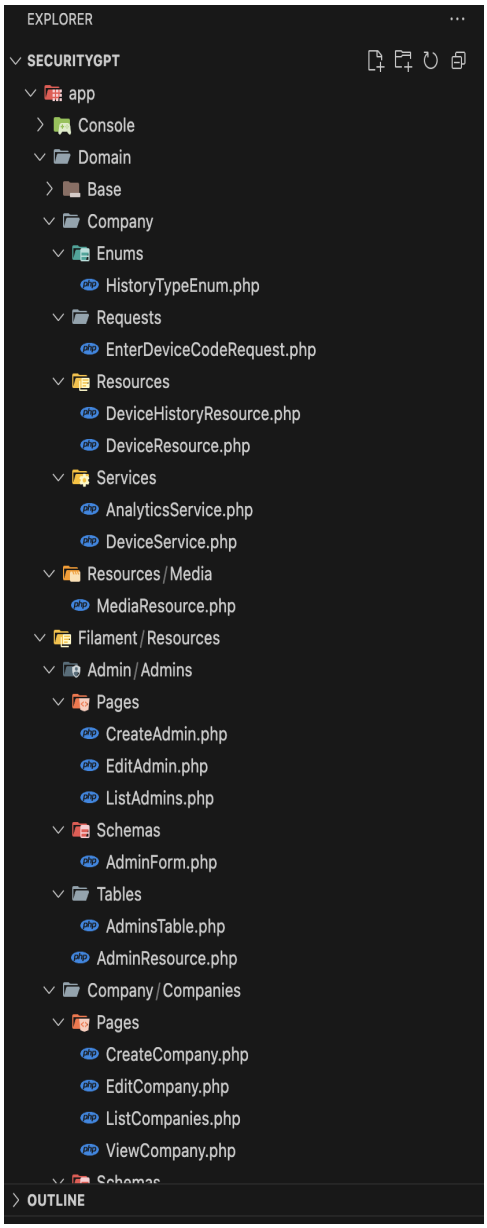

Рисунок 3.1 – Структурно-функціональна модель компонентів React-додатку

Коренева директорія містить основні папки, кожна з яких відповідає за окремий функціональний сегмент застосунку. Папка Assets використовується для зберігання статичних ресурсів, таких як стилі, шрифти та зображення, що необхідні для формування зовнішнього вигляду інтерфейсу. У папці Components розташовані всі повторно використовувані елементи, включаючи модульні компоненти, обгортки, кнопки та інAI частини інтерфейсу, що спрощують підтримку та повторне використання функціональних блоків. Директорія Context містить логіку керування глобальним станом застосунку. Контекст React дозволяє централізовано керувати змінними, які використовуються у різних частинах інтерфейсу, без потреби передавати їх через численні рівні вкладених компонентів. У папці Hook зберігаються кастомні хуки, що містять повторювану логіку, винесену в окремі модулі для підвищення читабельності та зменшення дублювання коду. Папка Page є однією з ключових, оскільки містить усі сторінки застосунку: сторінку авторизації, панель керування, сторінку перегляду пристроїв та інAI елементи інтерфейсу. Така організація забезпечує логічний поділ і полегшує роботу з маршрутизацією. У директорії Services зосереджені модулі для роботи з API, які відповідають за виконання HTTP-запитів, отримання відповідей від сервера та взаємодію з REST API. Централізація мережевої логіки запобігає її дублюванню та спрощує внесення змін. Папка Types використовується для зберігання типів, що застосовуються в проєкті, що забезпечує чіткість у визначенні структури об’єктів під час використання TypeScript.

Окрему роль у структурі застосунку відіграє сторінка DevicesPage, яка є центральним елементом для управління пристроями охоронної системи. На цій сторінці реалізовано механізм отримання та надсилання Bearer Token до серверної частини, що дозволяє завантажувати список пристроїв, отримувати інформацію про їхній стан та виконувати керуючі дії.

DevicesPage забезпечує обробку отриманих даних та їх відображення у зручному для користувача форматі, що робить її одним із ключових елементів взаємодії з веб-інтерфейсом охоронної системи (рисунок 3.2).

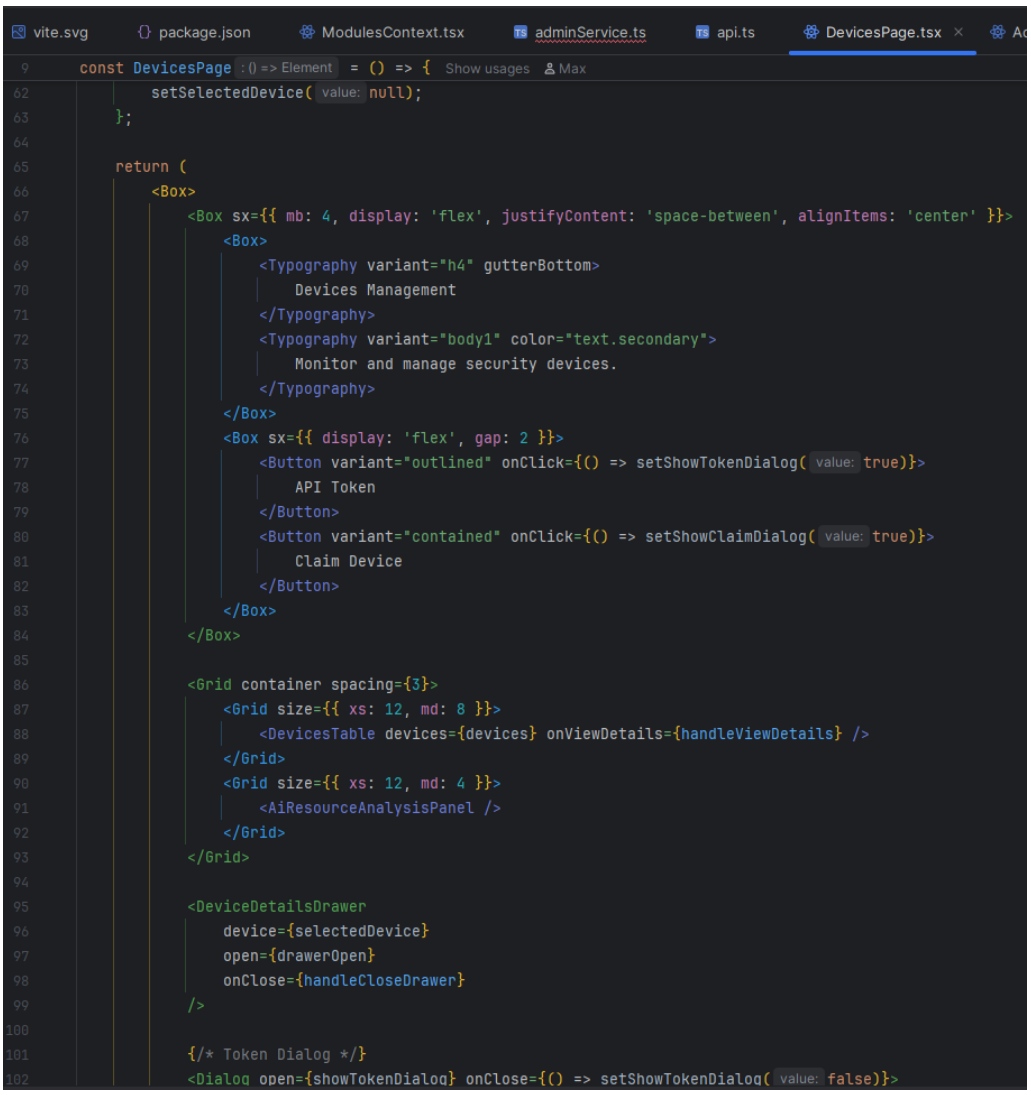

Рисунок 3.2 – Фрагмент програмного коду, що визначає структуру та компонування елементів React-компонента DevicesPage

Реалізована структура дозволяє підтримувати високу модульність коду та забезпечує чіткий розподіл відповідальності між елементами програмного забезпечення. Такий підхід значно спрощує процес тестування, оскільки кожен компонент можна перевіряти окремо, не зачіпаючи інших частин інтерфейсу. Крім того, модульність прискорює внесення змін, оскільки розробник має можливість локально змінювати функціональність, не порушуючи роботу інших модулів системи.

Важливою властивістю представленої структури є її масштабованість. Система здатна без ускладнень доповнюватися новими сторінками, сервісами або логікою, адже кожен структурний елемент винесено в окремий каталог та

має чітко визначену роль. Завдяки використанню окремих директорій для сторінок, компонентів та сервісів забезпечено підтримку принципу розділення відповідальності, що є базовою вимогою для побудови програмних систем середнього та високого рівня складності.

Значну роль у підтримці структури відіграє також використання TypeScript, який накладає чіткі правила на опис об’єктних структур, форм даних та параметрів, що функціонально взаємодіють між модулями. Зберігання типів у спеціальній папці спрощує інтеграцію нових частин системи та забезпечує узгодженість даних між клієнтською і серверною частиною.

Контекстна архітектура дає змогу уникнути дублювання логіки та забезпечує синхронність у відображенні інформації щодо статусу датчиків, журналів подій, результатів аналітики та користувацьких налаштувань. У свою чергу, кастомні хуки дають змогу винести повторювані операції – наприклад, обробку токенів, запитів чи системних подій – в окремі модулі, що підвищує рівень повторного використання коду.

Оцінюючи комплексно, розроблена в роботі структура проекту повністю відповідає вимогам до проєктування інтерфейсів безпеки, у яких критичними є надійність, передбачуваність та швидкість розробки. Розділення інтерфейсних компонентів, бізнес-логіки, мережевих сервісів і глобального стану дозволяє будувати систему, що здатна обробляти значні обсяги подій у реальному часі та інтегрувати нові функціональні модулі без ризику порушення роботи вже існуючих. Крім того, така організація дає можливість легко масштабувати проєкт у напрямі розвитку штучного інтелекту. Оскільки аналіз даних та інтелектуальні модулі також можуть бути винесені у власні сервіси та сторінки, структура дає змогу інтегрувати алгоритми класифікації спрацювань, виявлення аномалій або автоматичного формування звітів без необхідності змінювати фундаментальні елементи застосунку.

Тому можна вважати, що сформована структура проєкту не лише забезпечує підтримуваність та надійність веб-інтерфейсу охоронної системи, але й створює основу для подальшого розвитку системи, включаючи інтеграцію

додаткових модулів, алгоритмів штучного інтелекту та нових мережевих протоколів. У контексті кваліфікаційної роботи це є важливою практичною перевагою, оскільки демонструє можливість масштабування й адаптації системи до майбутніх вимог та розширених сценаріїв використання.

3.2 Розгортання веб-проекту на VPS-сервері

Процес розгортання розробленого вебзастосунку на промисловому сервері є ключовим етапом впровадження системи.

Для забезпечення контрольованого та ефективного середовища було обрано віртуальний приватний сервер VPS із встановленою операційною системою родини Linux та спеціалізованою панеллю керування FastPanel (рисунок 3.3), тобто, спочатку у DNS налаштовується запис, який вказує на IP вашого VPS.

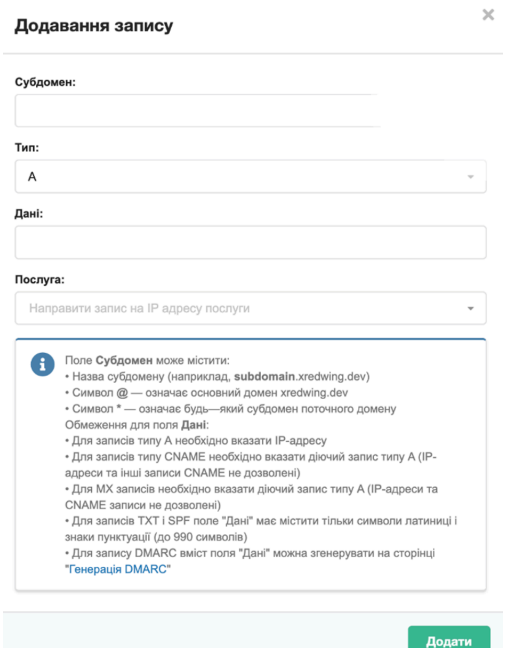

Рисунок 3.3 – Направлення субдомена на сервер

Розгортання починалось з підготовки інфраструктури домену та встановлення ОС. Після встановлення базової операційної системи на сервері

було інстальовано панель керування FastPanel, що значно прискорило подальше налаштування хостингу.

Далі на VPS встановлювали Ubuntu (рисунок 3.4), а потім FastPanel як панель керування, на зразок cPanel або ISPmanager, але безкоштовну.

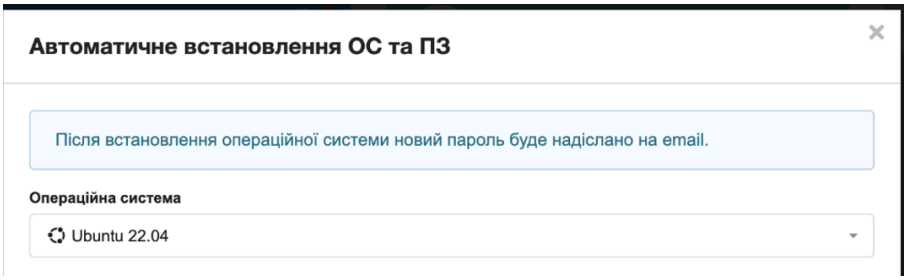

Рисунок 3.4 – Фрагмент встановлення операційної системи

FastPanel дозволяє не тільки створювати сайти, а і працювати з PHP, керувати базами даних, додавати SSL, створювати FTP/SSH доступ (рисунок 3.5).

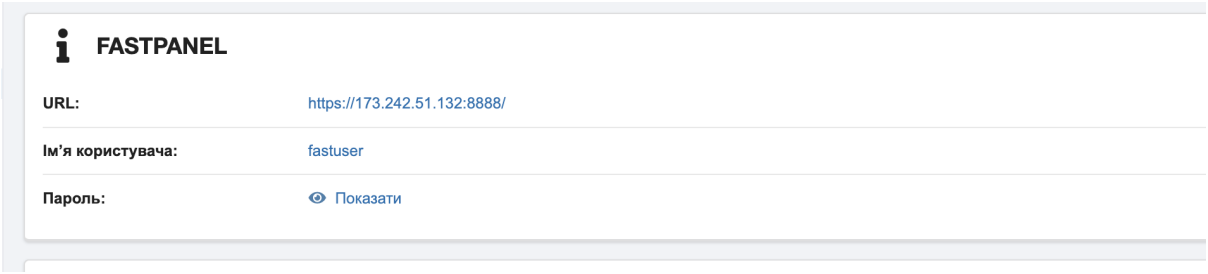

Рисунок 3.5 – Реєстрація у FastPanel

Конфігурація FastPanel відбувалась після первинного входу до FastPanel було введено електронну пошту для отримання безкоштовної ліцензії. У розділі налаштувань FastPanel була обрана необхідна версія PHP для глобального використання та конфігурації вебсервера.

Створення сайту і створення віртуального хосту здійснювалося вручну. Було вказано доменне ім'я та обрана версія PHP для бекенду, що забезпечило коректну роботу застосунку. Після створення сайту було згенеровано та завантажено облікові дані для доступу через SSH до користувача цього сайту.

Для коректної роботи фреймворку Laravel, що використовується для бекенду, необхідна специфічна конфігурація PHP та встановлення пакетного менеджера Composer.

Глобальне встановлення PHP та розширень відбувається через SSH, під обліковим записом суперкористувача root, було виконано встановлення необхідної версії PHP 8.2 та всіх обов'язкових розширень.

Це забезпечило повноцінну підтримку всіх функціональних можливостей фреймворку.

3.3 Розробка і проектування інтелектуального інтерфейсу користувача

Сторінка DevicesPage виконує функцію централізованого інструмента для моніторингу та керування пристроями охоронної системи та є ключовим елементом інтерфейсу системи. Вона забезпечує відображення загального списку пристроїв, а також надає можливість перегляду детальної інформації про кожен елемент у бічній панелі.

Основною зоною інтерфейсу є таблиця пристроїв, яка відображає перелік всіх підключених або керованих системою елементів.

На сторінці реалізовано функцію пошуку, що дозволяє здійснювати фільтрацію за ключовими параметрами, а також додано випадаючі списки для відбору пристроїв за типом та статусом.

Таблиця містить основні характеристики кожного пристрою, серед яких назва, тип, місцезнаходження, поточний статус, а також час останнього зв’язку з сервером і поле дій, що дозволяє виконувати керуючі команди або відкривати докладну інформацію.

Праворуч від таблиці розташована бічна панель, яка відкривається після вибору конкретного пристрою із загального списку. У цій панелі відображається розширена інформація про вибраний елемент, включаючи його ідентифікатор ID, назву, тип і поточний статус роботи.

Однією з ключових секцій бічної панелі є модуль аналізу використання ресурсів, у якому подано графічне відображення навантаження на пристрій за останню годину (рисунок 3.6).

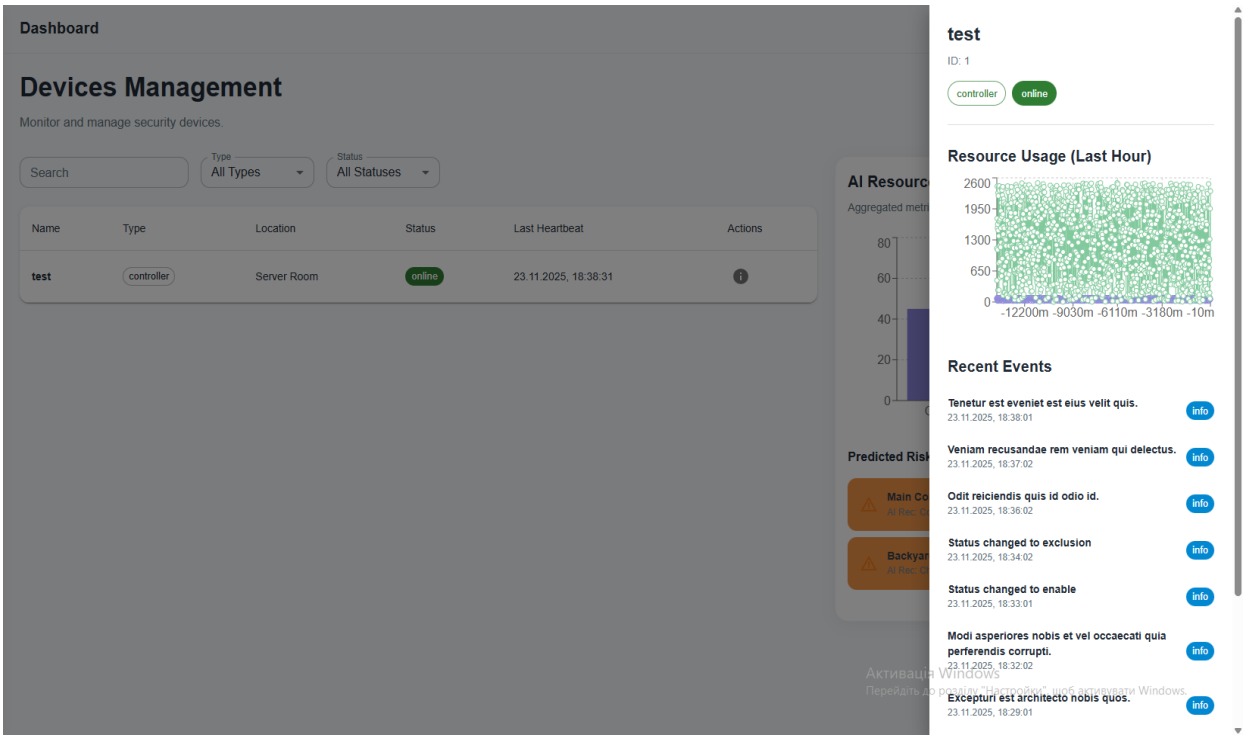

Рисунок 3.6 – Сторінка керування пристроями

Завдяки цьому блоку користувач може оцінити активність пристрою, частоту звернень і потенційне перевантаження. Секція останніх подій містить хронологічний перелік подій, пов’язаних із цим пристроєм, із зазначенням типу події, часу її виникнення та статусу, що дозволяє швидко оцінити роботу пристрою та виявити відхилення від норми.

Окремим функціональним елементом є блок прогнозованих ризиків, який відображає аналітику, сформовану системою штучного інтелекту. У цій секції подаються потенційні зони ризику або ділянки, де система прогнозує можливе порушення роботи. Цей елемент дозволяє завчасно виявляти небезпечні ситуації та оптимізувати процес реагування. Завдяки поєднанню таблиці загального огляду та детальної бічної панелі DevicesPage забезпечує повний контроль за станом пристроїв охоронної системи, надає оперативний доступ до технічної

інформації та інтегрує елементи аналітики та прогнозування, що підвищує ефективність управління системою.

Сторінка Security OS (рисунок 3.7) виконує роль головного дашборда системи безпеки та забезпечує узагальнений огляд ключових показників роботи всієї охоронної інфраструктури.

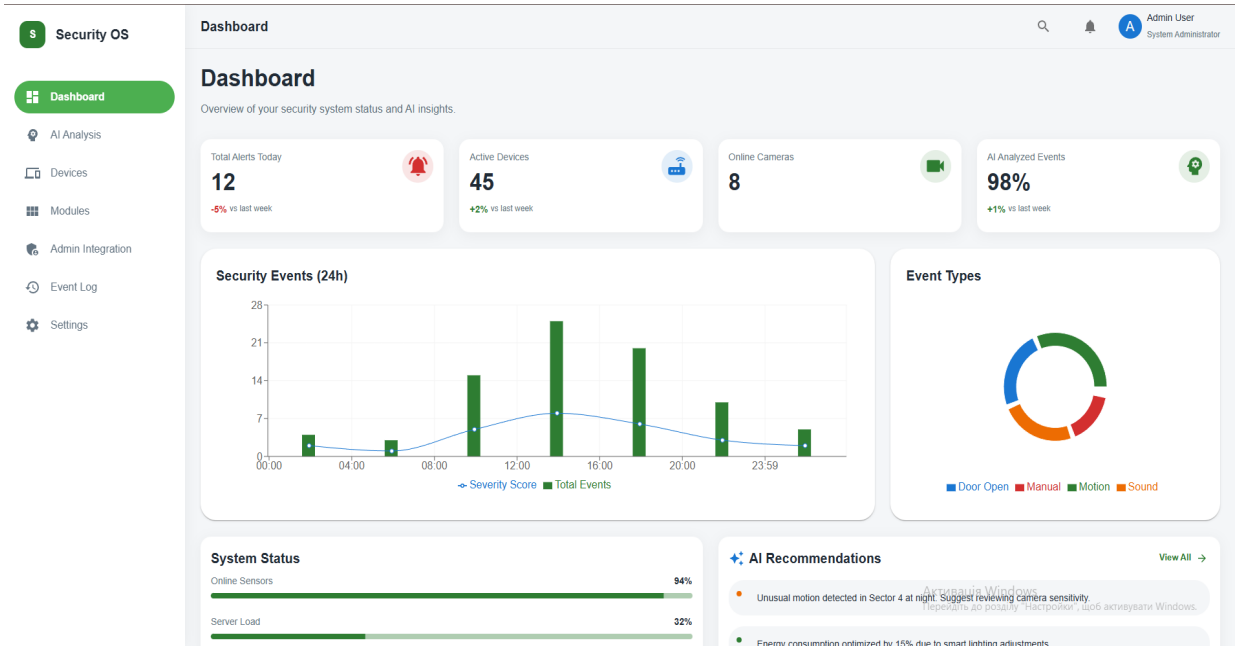

Рисунок 3.7 – Сторінка дашборду з елементами відтворення даних моніторингу системи

Інтерфейс цієї сторінки побудований таким чином, щоб користувач міг оперативно оцінити поточний стан системи, активність пристроїв, інтенсивність подій та аналітичні результати, сформовані модулем штучного інтелекту. Основним елементом є блок KPI-карток, побудованих на основі бібліотеки Recharts. Кожна картка відображає окремий агрегований показник, серед яких: загальна кількість сповіщень за день, кількість активних пристроїв, кількість камер, що перебувають у режимі онлайн, а також загальна кількість подій, оброблених штучним інтелектом. Ці показники дозволяють швидко оцінити поточне навантаження на систему та стан охоронних компонентів. У центральній частині дашборда розташовані графічні модулі, що відображають динаміку та

структуру подій. Графік «Security Events (24h)» поєднує лінійну та стовпчасту модель відображення даних для демонстрації інтенсивності подій за останню добу та їх часової концентрації. Це забезпечує можливість відстежувати періоди підвищеної активності та визначати часові закономірності виникнення критичних ситуацій. Додатковий графічний модуль «Event Types» представлений у вигляді кругової діаграми, яка демонструє розподіл подій за типами, такими як Door Open, Manual, Motion та Sound. Така візуалізація дає змогу оцінити, які типи подій є домінантними, та виявити потенційні точки ризику, пов’язані з певними видами активностей. Завершальним елементом є блок відображення статусу системи, що включає індикатори завантаження сенсорів і серверної частини. Цей модуль дозволяє оцінити рівень ресурсоємності системи, стабільність її роботи та наявність можливих технічних перевантажень. Сторінка «Security OS» об’єднує всі ключові аналітичні та інформаційні елементи, забезпечуючи комплексний огляд стану системи безпеки в одному інтерфейсі та дозволяючи користувачу ефективно контролювати ситуацію у режимі реального часу.

На рисунку 3.8 представлено інтерфейс модуля Devices Management програмної системи Security OS

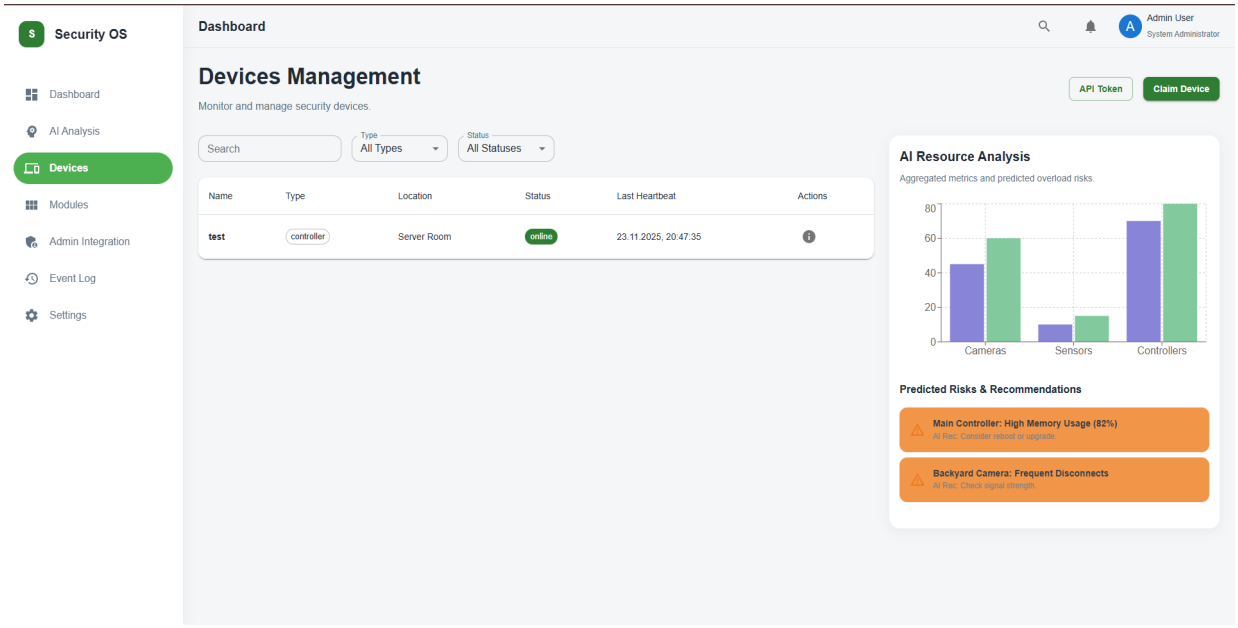

Рисунок 3.8 – Інтерфейс модуля керування пристроями в системі Security OS

Інтерфейс призначений для моніторингу та адміністрування підключених пристроїв безпеки. У центральній частині відображено таблицю з переліком пристроїв, де для кожного зазначено назву, тип, місце розташування, поточний статус та час останнього з'єднання. Ліва панель містить основне меню системи, що забезпечує доступ до модулів «Dashboard», «AI Analysis», «Devices», «Modules», «Admin Integration», «Event Log» та «Settings».

У правій частині інтерфейсу наведено блок AI Resource Analysis, який включає графічну візуалізацію завантаження ресурсів різних категорій пристроїв таких, як камери, сенсори, контролери та секцію Predicted Risks & Recommendations із прогнозами ризиків та рекомендаціями системи штучного інтелекту щодо можливих збоїв або необхідності оптимізації.

Інтерфейс також містить елементи керування, такі як кнопки API Token та Claim Device, а у верхньому правому куті відображено дані авторизованого адміністратора. Цей інтерфейс забезпечує наочний, структурований та зручний для користувача спосіб керування інфраструктурою системи безпеки та контролю за її станом у режимі реального часу.

На рисунку 3.9 представлено вигляд панелі деталізації пристрою в системі Security OS, яка відкривається при виборі конкретного елемента зі списку пристроїв.

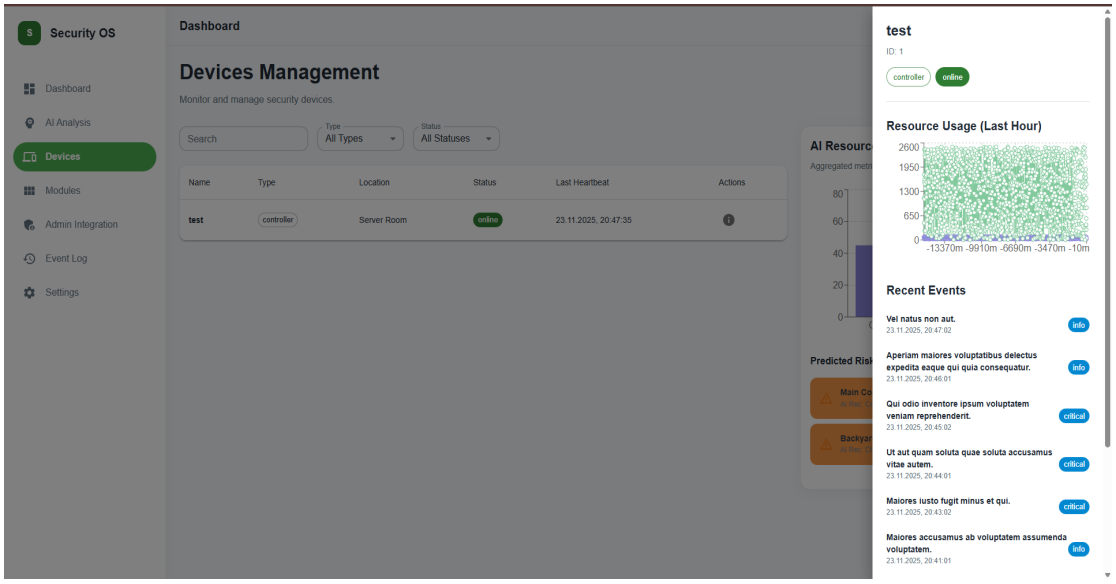

Рисунок 3.9 – Панель деталей пристрою в системі Security OS

Дана панель забезпечує детальний моніторинг стану окремого пристрою, аналіз його ресурсного навантаження та швидкий доступ до журналу подій, що є важливим для підтримки безперервного функціонування системи безпеки

В усіх вікнах панелі у правій частині екрана відображаються основні атрибути вибраного пристрою: його назва, ідентифікатор, тип та поточний статус.

Наприклад, є ще вікно Resource Usage, де графік містить візуалізацію використання ресурсів пристрою за останню годину у вигляді точкової діаграми, що дозволяє оцінити характер навантаження та можливі пікові значення. У секції Recent Events представлено хронологічний перелік останніх подій, пов’язаних із роботою пристрою: інформаційні та критичні повідомлення. Тобто, кожен запис містить короткий опис події та точний час її виникнення, що забезпечує можливість оперативного аналізу інцидентів та оцінки стабільності роботи обладнання.

Блок AI Recommendations є окремим елементом дашборда системи. Він виконує функцію відображення рекомендацій, зауважень або виявлених аномалій, сформованих модулем штучного інтелекту.

У цьому блоці подається структурований список висновків, які AI отримує в результаті аналізу даних із сенсорів, журналів подій, ресурсного навантаження та поведінкових патернів пристроїв.

Кожен пункт рекомендацій містить короткий опис виявленої ситуації, її потенційну критичність та можливі дії, які доцільно виконати для усунення проблеми або запобігання ризикам.

На рисунку 3.10 наведено фрагмент запиту до клієнтського модуля api.ts, який реалізує налаштування HTTP-клієнта Axios для забезпечення безпечної взаємодії веб-додатка з серверною частиною. Представлений фрагмент коду є важливою частиною інформаційної інфраструктури системи, оскільки забезпечує захищеність обміну даними, контроль доступу, керованість і надійність взаємодії з API.

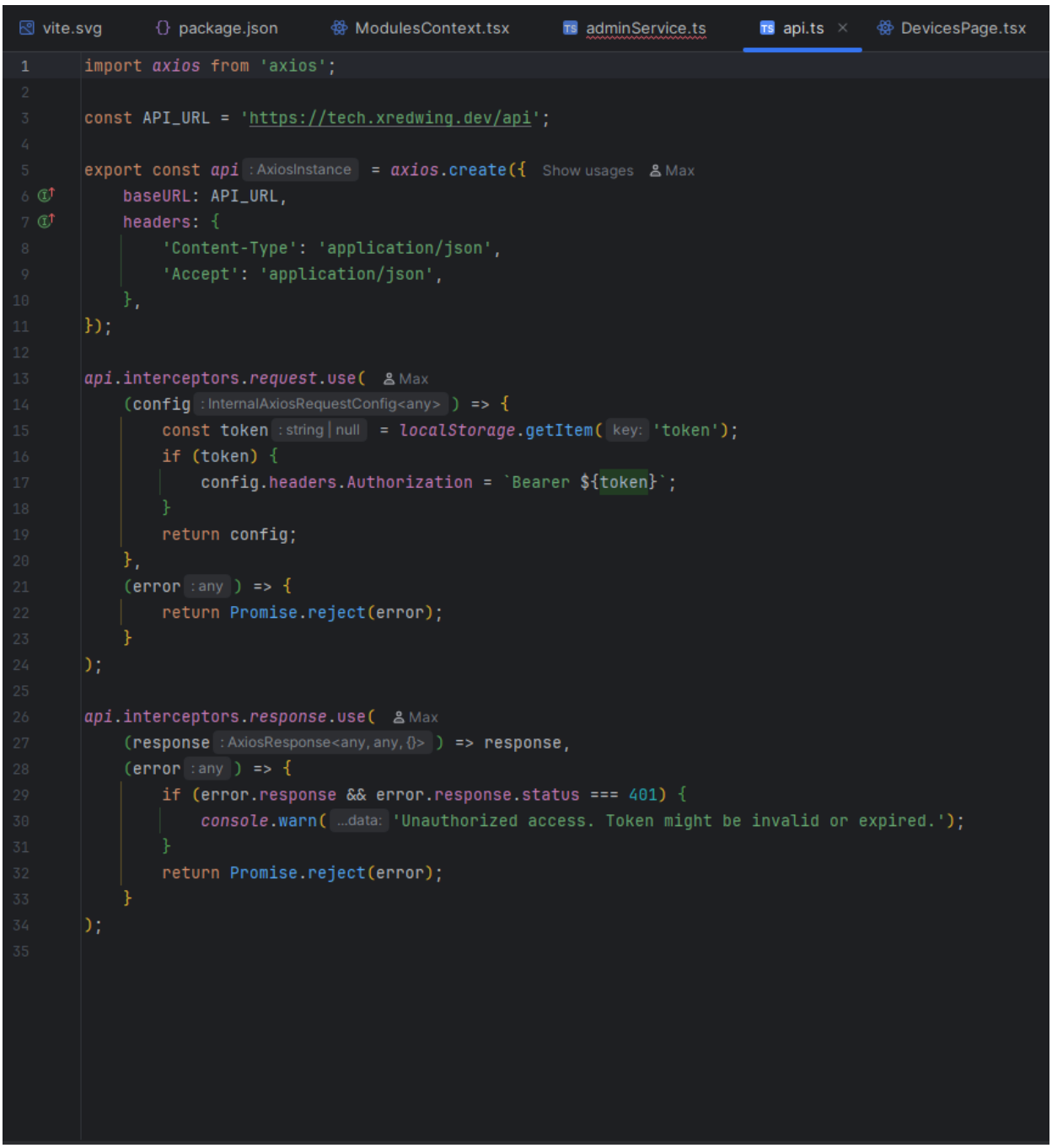

Рисунок 3.10 – Фрагмент модуля клієнтсько-серверної взаємодії з використанням Axios-інтерцепторів для авторизації

Такий підхід дозволяє своєчасно реагувати на потенційні загрози та підвищує загальну ефективність роботи охоронної системи, забезпечуючи превентивний аналіз ситуацій, що можуть вплинути на рівень безпеки.

Код містить визначення базової URL-адреси API, конфігурацію заголовків запитів, а також два ключові інтерцептори – для вихідних і для вхідних HTTP-відповідей.

Виявлені аномальні критичні стрибки графіків можуть стосуватися підозрілої активності датчиків, нестандартної поведінки окремих пристроїв, перевищення допустимих показників навантаження чи повторюваних подій, які виходять за межі стандартного сценарію.

3.4 Взаємодія клієнтської та серверної частини

Файл Api.ts використовується як центральний модуль для роботи з авторизацією та взаємодією з сервером, у ньому зосереджено всю логіку логінізації та авторизації користувача.

У цьому файлі задається константа API_URL, яка містить базову адресу серверного API і використовується для формування всіх подальших запитів до бекенду.

На основі цієї адреси створюється об’єкт API, що слугує окремим екземпляром клієнта для HTTP-запитів і зберігає всі налаштування, необхідні для коректної роботи з сервером, зокрема базову URL-адресу, тип контенту, тайм-аути та інAI параметри.

Для обробки авторизаційних даних у цьому модулі використовується механізм перехоплення запитів через метод api.interceptors.request.use, за допомогою якого перед відправленням кожного запиту виконується додаткова обробка.

На цьому етапі в localStorage зберігається або оновлюється токен доступу, отриманий після успішної авторизації, а також при формуванні запиту токен додається до заголовків.

Це дає змогу забезпечити автоматичну передачу авторизаційних даних для всіх захищених endpoint-ів без необхідності вручну вказувати токен у кожному окремому виклику, а також централізовано керувати процесом авторизації в межах одного файлу.

Така організація Api.ts спрощує підтримку коду, робить логіку взаємодії з сервером передбачуваною та забезпечує єдиний підхід до обробки запитів і авторизаційних даних (рисунок 3.11).

```
getMyDevices: async (): Promise<Device[]> => {
    const response :AxiosResponse<any, any, {}>  = await api.get( url: '/company/my_device');
    console.log( ...data: 'getMyDevices response:', response.data);

    let rawDevices: any[] = [];
    if (Array.isArray(response.data)) {
        rawDevices = response.data;
    } else if (response.data && Array.isArray(response.data.data)) {
        rawDevices = response.data.data;
    } else if (response.data && Array.isArray(response.data.devices)) {
        rawDevices = response.data.devices;
    }

    return rawDevices.map((d: any) => ({
        id: String(d.id),
        name: d.name || 'Unknown Device',
        type: 'controller', // Default fallback
        location: 'Server Room', // Default fallback
        status: 'online', // Default fallback
        lastHeartbeat: new Date().toISOString(),
    }));
},
```

Рисунок 3.11 – Приклад обробки гетерогенних структур даних під час отримання списку пристроїв із зовнішнього API

Далі продемонстровано механізм формування асинхронного запиту до серверного інтерфейсу з метою отримання телеметричних даних конкретного пристрою.

Тут адаптивно інтерпретуються вхідні дані, враховуючи різні формати вкладеності, після чого виконує хронологічне впорядкування записів.

На основі журналів генерується історія температурних та енергетичних показників, а також формується вибірка подій безпеки з класифікацією рівнів критичності та контекстуальним описом.

Підхід, що забезпечує узгодженість телеметричних даних, підвищує точність подальшого аналізу та створює єдину модель даних для аналітичних модулів системи представлено на рисунку 3.12.

```
getDeviceMetrics: async (deviceId: string): Promise<DeviceHistory> => {
    const response : AxiosResponse<any, any, {}> = await api.get( url: `/company/my_device/${deviceId}`);
    console.log( ...data: 'getDeviceMetrics response:', response.data);

    let logs: any[] = [];
    if (response.data && Array.isArray(response.data.data)) {
        logs = response.data.data;
    } else if (Array.isArray(response.data)) {
        logs = response.data;
    }

    logs.sort((a : any , b : any ) => new Date(a.created_at).getTime() - new Date(b.created_at).getTime());

    const cpuHistory : any[] = logs.map(log : any => log.temperature_c || 0);

    const memoryHistory : any[] = logs.map(log : any => log.instant_power_watts || 0);

    const events: SecurityEvent[] = logs
        .filter(log : any => log.status === 'error' || log.status === 'exclusion' || log.status === 'enable')
        .map(log : any => ({
            id: String(log.id),
            timestamp: log.created_at,
            type: 'system' as const,
            severity: (log.status === 'error' ? 'critical' : 'info') as 'critical' | 'info' | 'warning',
            deviceId: String(log.device_id),
            deviceName: 'Device ' + log.device_id,
            description: log.message || `Status changed to ${log.status}`,
            location: 'Unknown',
        }))
        .reverse()
        .slice( start: 0, end: 10);

    return {
        cpuHistory: cpuHistory.length > 0 ? cpuHistory : [20, 25, 30, 45, 40, 35],
        memoryHistory: memoryHistory.length > 0 ? memoryHistory : [50, 52, 55, 60, 58, 55],
        events: events,
    };
},
```

Рисунок 3.12 – Приклад реалізації запиту для отримання телеметрії окремого пристрою

З наведених програмних фрагментів видно, що взаємодія клієнтської частини системи з API здійснюється за допомогою HTTP-запитів, через які інтерфейс отримує дані від серверної частини та надсилає необхідні команди. Клієнтський застосунок формує та відправляє запити на визначені endpoint-и серверного API, використовуючи попередньо налаштований клієнт у файлі

Api.ts. Після авторизації користувача на стороні клієнта зберігається токен доступу, який автоматично додається до кожного запиту за допомогою механізму перехоплення. Це забезпечує доступ до захищених ресурсів API без потреби повторного введення даних.

У відповідь на запити система отримує структуровані дані у форматі JSON, які інтерфейс обробляє та відображає у відповідних компонентах. Наприклад, для сторінки DevicesPage клієнт надсилає запит на отримання переліку пристроїв, після чого отримана інформація використовується для формування таблиці, бічної панелі та аналітичних блоків. Аналогічним чином дашборд Security OS звертається до API для отримання статистики, аналітичних даних та AI-висновків.

Уся логіка роботи з API зосереджена у спеціальних сервісних модулях, що відповідають за формування запитів, обробку відповідей і передачу результатів на рівень компонентів. Такий підхід забезпечує чіткий розподіл відповідальності, спрощує підтримку коду та робить взаємодію з API передбачуваною та стабільною в межах усієї системи.

Система охоронного моніторингу передбачає наявність адміністративної панелі, у якій користувач отримує повний контроль над керуванням пристроями. У адмін-панелі користувач має можливість реєструвати нові пристрої, вказуючи їх технічні параметри та місце встановлення. Після реєстрації кожному пристрою автоматично присвоюється унікальний токен доступу, який використовується для підключення до системи та автентифікації запитів. Завдяки цьому токену будь-який сумісний пристрій може бути доданий у систему без потреби виконання ручних конфігурацій.

Після підключення пристрою система забезпечує постійний обмін даними між ним і серверною частиною. Пристрій передає інформацію про свій стан, активність, рівень навантаження та зареєстровані події. На основі отриманих даних система формує деталізовані звіти, які містять історію роботи пристрою, критичні події, статистику активності та результати аналізу, виконаного модулем штучного інтелекту. Такі звіти можуть використовуватися як для

оперативного моніторингу, так і для довгострокового аналізу функціонування обладнання.

Адмін-панель також дозволяє керувати статусом пристроїв, виконувати віддалені дії, отримувати прогнози потенційних ризиків, переглядати результати AI-аналізу та налаштовувати правила взаємодії пристроїв із системою. Завдяки цьому забезпечується повноцінна централізована інфраструктура, що дозволяє об’єднати велику кількість сенсорів і модулів у єдине кероване середовище.

Система охоронного моніторингу передбачає наявність адміністративної панелі, у якій користувач отримує повний контроль над керуванням усіма підключеними пристроями. У адмін-панелі користувач може реєструвати нові пристрої, задаючи їх технічні параметри, функціональні можливості та місце встановлення. Після додавання кожному пристрою автоматично генерується унікальний токен доступу, який використовується для автентифікації та безпечного підключення до системи. Завдяки цьому токену будь-який сумісний пристрій може бути інтегрований без додаткових ручних конфігурацій.

Після підключення пристроїв система забезпечує постійний обмін даними між ними та серверною частиною. Кожен пристрій надсилає інформацію про стан, події, рівень навантаження та телеметрію, необхідну для комплексного моніторингу. На основі цих даних формуються деталізовані звіти, що містять історію роботи обладнання, критичні події, статистику активності та інAI технічні показники.

Окремим елементом функціонування системи є постійна взаємодія з моделями штучного інтелекту OpenAI. Система регулярно звертається до AI-модуля, надсилаючи агреговані дані датчиків, журнали подій, статистичні показники та іншу службову інформацію. Модель OpenAI здійснює аналіз отриманих даних, виявляє аномалії, визначає нетипові патерни у поведінці пристроїв, оцінює рівень навантаження та можливі ризики. На основі цього аналізу AI формує рекомендації щодо оптимізації роботи пристроїв, зменшення навантаження, покращення конфігураційних параметрів або усунення

потенційних проблем. Рекомендації передаються назад у систему та відображаються на відповідних сторінках інтерфейсу.

Адмін-панель також надає можливість керувати статусом пристроїв, запускати віддалені дії, переглядати прогнозовані ризики, отримувати результати AI-аналізу та налаштовувати правила взаємодії елементів системи. Завдяки цьому формується централізована інфраструктура, яка поєднує велику кількість пристроїв, забезпечує повноцінний моніторинг у режимі реального часу та використовує інтелектуальні механізми для підвищення ефективності та надійності охоронної системи.

3.5 Аналітика подій та модуль моніторингу

Процес тестування системи був спрямований на перевірку коректності роботи всіх її компонентів, оцінку стабільності функціонування, виявлення можливих помилок та підтвердження відповідності системи вимогам.

Оскільки тестування охоплювало як клієнтську частину, так і серверну логіку, а також взаємодію з API, роботу AI-модуля та передачу даних між окремими пристроями, то найперше було виконано модульні тести, які дозволили перевірити правильність роботи окремих функцій, сервісів та компонентів інтерфейсу. Спочатку було приділено увагу перевірці механізмів авторизації, обробки токенів, формування запитів до API та обробки відповідей. Наступним етапом стало інтеграційне тестування, під час якого перевірялася взаємодія між клієнтською частиною, сервером та базою даних.

Окремо оцінювалася коректність передачі даних між пристроями й системою, зокрема передавання телеметрії, оновлення статусів, формування журналів подій та відображення цих даних у веб-інтерфейсі. Важливою частиною інтеграційних перевірок була оцінка роботи механізмів маршрутизації та відображення інформації на сторінках DevicesPage та Security OS. Окремий блок тестування був присвячений перевірці функціонування модуля штучного інтелекту.

Для цього була використана система Telescope, яка забезпечила можливість детального моніторингу виконання запитів, аналізу серверних відповідей та відстеження взаємодії між окремими компонентами в режимі реального часу (рисунок 3.13).

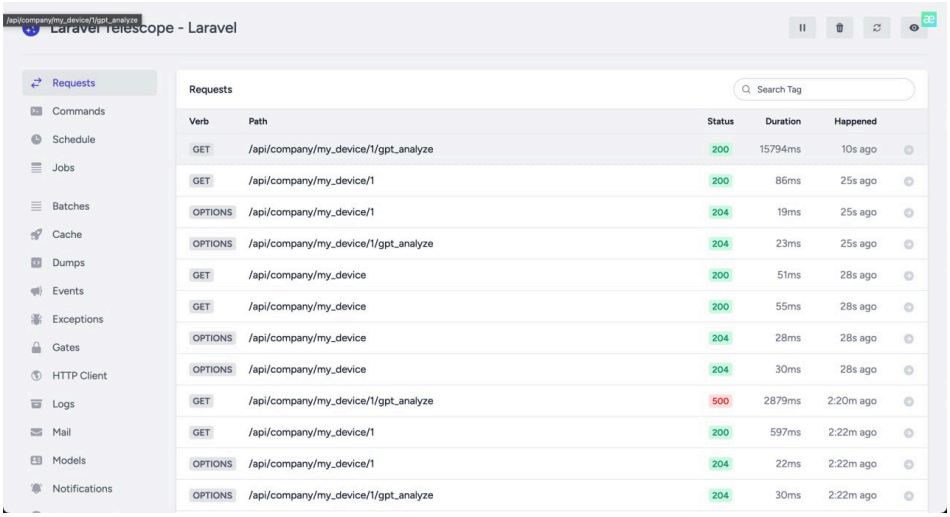

Рисунок 3.13 – Перевірка статусу з’єднань в системі Telescope

Інструмент дав змогу фіксувати параметри запитів, часові характеристики їх оброблення, журнал подій, а також можливі аномалії або відхилення у поведінці сервісів (рисунок 3.14).

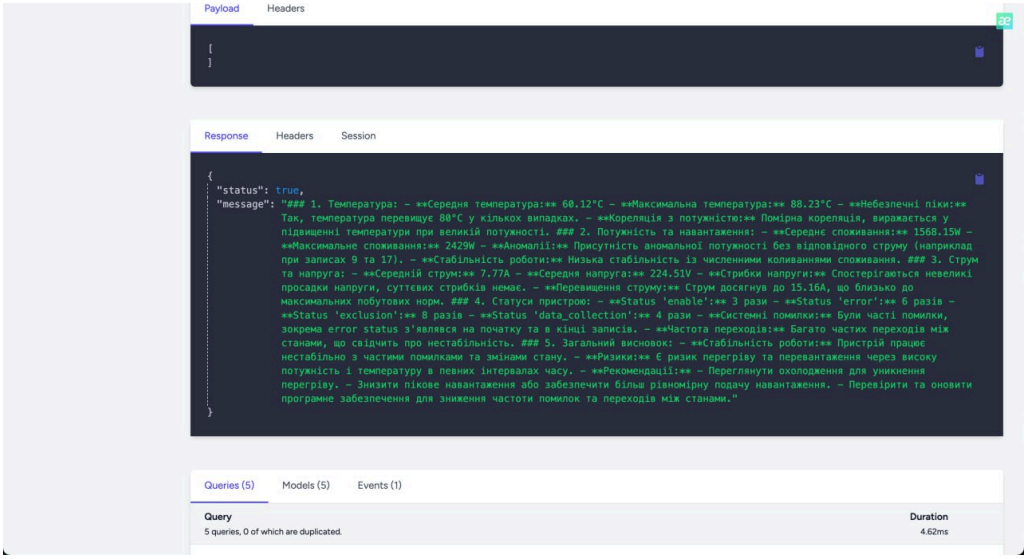

Рисунок 3.14 – Результати тестування системних модулів за допомогою платформи Telescope

Завдяки цьому стало можливим виявлення некоректних або нестабільних сценаріїв, зокрема під час обробки телеметричних даних, формування аналітичних повідомлень і передачі внутрішньосистемних подій. Отримані результати сприяли точному локалізуванню потенційних помилок, оптимізації логіки взаємодії між сервісами та підвищенню надійності роботи AI-модуля й системи загалом.

Також проводилося порівняння фактичних подій із AI-аналізом, оцінювалися точність виявлення аномалій, коректність формування рекомендацій та здатність моделі обробляти великі обсяги даних за певний період часу. Це дозволило переконатися, що система AI Recommendations надає обґрунтовані висновки та придатна до використання у реальних умовах. Після цього було проведено навантажувальне тестування, спрямоване на визначення стійкості системи при зростанні кількості пристроїв, частоти подій та обсягу запитів до серверної частини.

У ході тестів оцінювалися швидкість реакції інтерфейсу, час генерації графіків, стабільність роботи API та поведінка системи у пікові моменти. Завершальним етапом стала перевірка користувацьких сценаріїв, яка включала моделювання реальної взаємодії: реєстрацію пристроїв, їх авторизацію через токени, перегляд аналітики, формування звітів, виконання керуючих дій та взаємодію з адмін-панеллю. Цей етап дозволив оцінити зручність використання системи та відповідність її функцій очікуваній логіці роботи. У результаті тестування було підтверджено працездатність системи, виявлено та усунено похибки, оптимізовано роботу окремих модулів та підвищено стабільність взаємодії між компонентами, що забезпечило готовність системи до подальшого впровадження.

Зокрема, протестували набір даних, які відображають активність охоронної системи протягом 24 годин.

На рисунку 3.15 подано зміну активності охоронної системи приміщення протягом доби на основі залежності між кількістю зафіксованих подій та індексом їхньої критичності.

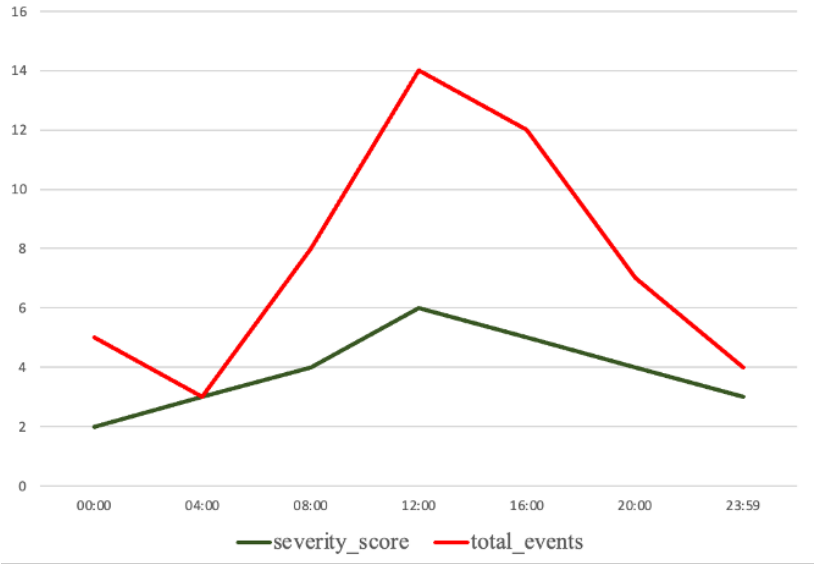

Рисунок 3.15 – Динаміка кількості подій та індексу критичності системи протягом доби

Графік демонструє, що у нічний період спостерігається відносно невелика кількість подій із низьким рівнем небезпеки, що відповідає загальній пасивності середовища. У ранкові години обидва показники зростають, що може бути пов’язано з підвищенням рухової активності та зовнішніх впливів на систему. Максимальні значення зафіксовано у середньоденний проміжок, коли кількість подій та їх критичність досягають піку, що вказує на найбільше навантаження на систему у цей час. Подальший часовий відрізок характеризується поступовим зменшенням інтенсивності подій та зниженням їх критичності, що відображає стабілізацію умов. Увечері активність знову зменшується, а події стають менш небезпечними, що корелює із завершенням робочих процесів. Така динаміка підтверджує коректність функціонування системи та дозволяє визначити часові інтервали, у яких необхідне посилене технічне чи аналітичне забезпечення.

На рисунку 3.16 наведено порівняльну характеристику двох ключових параметрів роботи охоронної системи: кількості сенсорів, що перебувають у робочому стані, та поточного навантаження на серверну інфраструктуру. Дані демонструють, що більАІсть сенсорів системи функціонують стабільно, що підтверджується високим значенням показника активності.

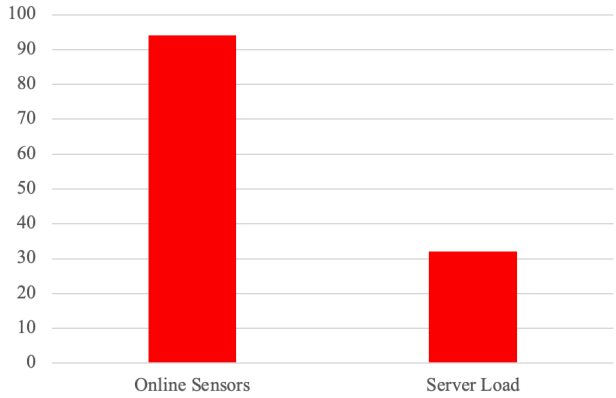

Рисунок 3.16 – Співвідношення кількості активних сенсорів та навантаження на сервер системи

Водночас навантаження на сервер є суттєво нижчим, що свідчить про достатній запас обчислювальних ресурсів та ефективність алгоритмів оброблення даних. Таке співвідношення є індикатором оптимальної взаємодії між периферійними пристроями та центральним сервером, а також підтверджує здатність системи забезпечувати безперервний моніторинг без ризику перевантаження інфраструктури.

На діаграмі 3.17 подано статистичні показники активності підсистем охоронної системи за визначений період.

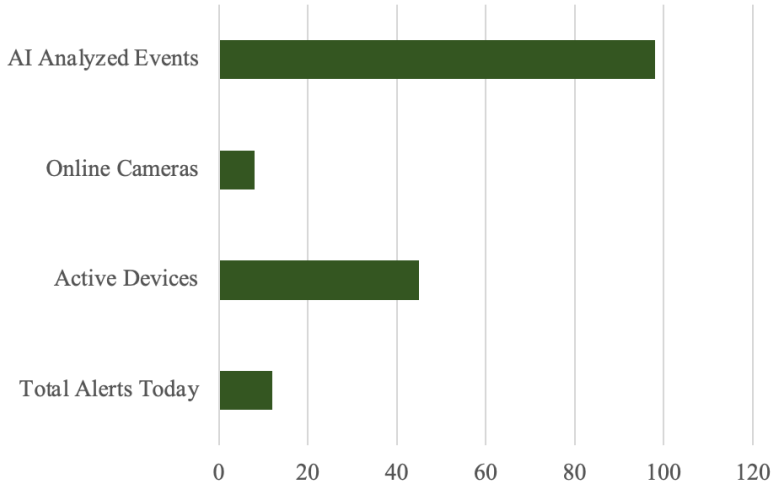

Рисунок 3.17 – Діаграма показників активності системи відеоспостереження та штучного інтелекту

Дані дають можливість оперативно оцінити поточну активність системи, включно з кількістю отриманих сповіщень, числом задіяних у роботі пристроїв, кількістю камер, що перебувають у мережевому доступі, а також часткою подій, які були успішно інтерпретовані алгоритмами штучного інтелекту.

Найбільше значення демонструє кількість подій, опрацьованих алгоритмами штучного інтелекту, що свідчить про інтенсивну роботу модулів автоматичного аналізу відеопотоку та сенсорних даних. Показник активних пристроїв перебуває на другому місці та демонструє масштаб залученного обладнання, що бере участь у моніторингу об’єкта. Кількість онлайн-камер є значно нижчою, що може пояснюватися специфікою доступних конфігурацій або тим, що окремі пристрої не використовують відеопотік, а працюють як сенсори. Загальна кількість тривожних подій за день залишається порівняно невисокою, що може свідчити про відсутність критичних ситуацій або про ефективність алгоритмів відфільтровування хибних сповіщень.

Таким чином, діаграма забезпечує узагальнений огляд стану системи в реальному часі та дозволяє оперативно оцінити інтенсивність роботи інфраструктури.

Проведене тестування підтвердило працездатність розробленої охоронної системи, узгодженість роботи її компонентів та відповідність функціональних характеристик поставленим вимогам. Аналіз результатів показав стабільність серверної частини, надійність взаємодії між клієнтським застосунком, API та модулем штучного інтелекту, а також здатність системи обробляти зростаючі обсяги даних без істотного зниження продуктивності. Візуальні дані дашборду та отримані журнали підтверджують коректне відображення подій, ефективність механізмів аналітики, відсутність критичних вузьких місць та достатній запас ресурсів для масштабування.

Сукупність проведених випробувань дозволяє зробити висновок про готовність системи до практичного застосування, подальшого впровадження та розширення функціональних можливостей.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було вирішено усі поставлені завдання.

Проаналізовано сучасний стан розвитку охоронних систем та порівняно їхні ключові технології, переваги та обмеження. Виявлено, що сучасні системи значно розширюють функціональні можливості завдяки інтеграції штучного інтелекту, проте обмеження виникають через високу вартість впровадження, потребу у спеціалізованому обладнанні та складність обробки великих обсягів даних. Також помічено, що ринок безпекових рішень активно розвивається у напрямку автоматизації та прогнозного аналізу подій.

Досліджено технологічні підходи до розробки веб-інтерфейсів і організації взаємодії з REST API, враховуючи питання модульності, масштабованості та тестування цілісності систем. Виявлено, що застосування компонентного підходу та гібридної архітектури REST і WebSocket забезпечує гнучкість, швидкість обробки даних у реальному часі та зручність масштабування. Крім того, тестування цілісності на рівнях апаратного, програмного та інформаційного забезпечення дозволяє підвищити надійність та безпеку системи.

Проаналізовано сучасні інструменти та методи розробки програмних компонентів, включно з використанням мовних моделей штучного інтелекту та фреймворку React Native для створення клієнтської частини системи. Виявлено, що використання React Native дозволяє швидко створювати кросплатформні мобільні інтерфейси з високою інтерактивністю, а інтеграція AI-моделей значно покращує аналітичні можливості системи, зокрема у розпізнаванні образів, прогнозуванні подій та класифікації аномалій.

Розроблено структуру та архітектуру охоронної системи, включаючи проектування інтерфейсу користувача, налаштування серверної частини та організацію взаємодії між клієнтом і сервером. Виявлено, що чітка ієрархія компонентів, централізація логіки взаємодії з API та використання типізації

TypeScript забезпечують зрозумілість коду, спрощують підтримку та модернізацію системи, а також дозволяють ефективно обробляти великі обсяги даних у реальному часі.

Проведено тестування функціональності, стабільності та безпеки розробленої системи, включаючи модулі аналітики та моніторингу подій. Виявлено, що система демонструє високу точність обробки даних, стійкість до помилок і надійність у роботі з різними сценаріями подій, а аналітичні модулі забезпечують ефективний контроль стану об’єктів системи і швидке виявлення потенційних загроз.

Отже, мету роботи досягнуто, а розроблена система успішно виконує функції моніторингу, аналітики та управління охоронними об’єктами. Подальше вдосконалення системи може передбачати інтеграцію більш точних моделей штучного інтелекту для прогнозування подій, розширення підтримки різних типів сенсорів і пристроїв, впровадження автоматизованих сповіщень у реальному часі та оптимізацію архітектури для обробки ще більших обсягів даних.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Христинець Н. А., Биков С. О., Марчук О. Ю. Прототипи інтелектуальних систем безпеки і управління інфраструктурою на базі IoT-платформ. *Комп'ютерно-інтегровані технології: освіта, наука, виробництво.* 2025. №60. С. 63-69.

2. Xia J. Hikvision Price Prediction based on ARIMA, OLS Multiple Regression and Random Forest. *Highlights in Science, Engineering and Technology*. 2024. Vol. 88. P. 546-552.

3. Improved KNN-based Stock Price Prediction. *Academic Journal of Computing & Information Science*. 2024. Vol. 7, no. 6. P. 91-112.

4. UK turns against Chinese Hikvision surveillance cameras. *Biometric Technology Today*. 2022. Vol. 2022, no. 4. P. 423-440.

5. Rick Mullin. AI developer Nvidia invests in Recursion. *C&EN Global Enterprise*. 2024. Vol. 101, no. 23. P. 8-9.

6. NVIDIA Launches Affordable Gen AI «Supercomputer». *The Engineer*. 2025. Vol. 302, no. 7963. P. 40-59.

7. Tencent A. Deep multiple instance learning-enabled gene mutation prediction of lung cancer from histopathology images. 2024. Vol. 7, no. 6. P. 123-137.

8. I-ViSE: Interactive Video Surveillance as an Edge Service using Unsupervised Feature Queries / S. Y. Nikouei et al. *IEEE Internet of Things Journal*. 2023. P. 1. URL: https://doi.org/10.1109/jiot.2023.3016825 (date of access: 19.07.2025).

9. Губчакевич С. А., Куперштейн Л. М. Система охоронно-тривожної сигналізації. Вінниця: ВНТУ, 2021. 211 с.

10. Duong H.-T., Le V.-T., Hoang V. T. Deep Learning-Based Anomaly Detection in Video Surveillance: A Survey. *Sensors*. 2023. Vol. 23, no. 11. P. 50-64.

11. Zhang W., Lazaro J. Survey on Network Security Traffic Analysis and Anomaly Detection Techniques. *International Journal of Emerging Technologies and Advanced Applications*. 2024. Vol. 1, no. 4. P. 8-16.

12. An Analysis of Artificial Intelligence Techniques in Surveillance Video Anomaly Detection: A Comprehensive Survey / E. Şengönül et al. *Applied Sciences*. 2023. Vol. 13, no. 8. P. 49-56.

13. Singh R., Gill S. S. Edge AI: A survey. *Internet of Things and Cyber-Physical Systems*. 2023. URL: https://doi.org/10.1016/j.iotcps.2023.02.004 (date of access: 23.07.2025).

14. Kuchuk H., Malokhvii E. Integration of IOT with Cloud, Fog and Edge Computing: a Review. *Advanced Information Systems*. 2024. Vol. 8, no. 2. P. 65-78.

15. Privacy-Preserving Video Anomaly Detection: A Survey / Y. Liu et al. *IEEE Transactions on Neural Networks and Learning Systems*. 2025. P. 1-22. URL: https://doi.org/10.1109/tnnls.2025.3600252 (date of access: 23.07.2025).

16. Alioanei C., Popescu N. AI-Based Solutions for Security and Resource Optimization in IoT Environments: A Systematic Review. *Information*. 2025. Vol. 16, no. 10. P. 841.

17. Kwak N., Lee B. Deep Learning Applied Abnormal Human Behavior Detection in Video Surveillance Systems - A Survey. *International JOURNAL OF CONTENTS*. 2024. Vol. 20, no. 4. P. 84-95.

18. Survey: Anomaly Detection In Surveillance Videos / E. A. Mahareek et al. *International Journal of Theoretical and Applied Research*. 2024. Vol. 3, no. 1. P. 328-342.

19. Волонтир Л., Потапова Н. Інтелектуальний аналіз даних оцінки матеріальних потоків логістичної системи. *Наука і техніка сьогодні*. 2022, №4. 124 с.

20. Терлецький Т. В., Федорчук-Мороз В. І., Кайдик О. Л. Системи пожежної сигналізації: підручник. Луцьк, 2022. 130 с.

21. J. Pradeep. Efficient Vehicle Detection System using OCR and REST API. *International Journal of Engineering Research and*. 2020. Vol. V9, no. 07.

22. Edge Computing and Cloud Computing for Internet of Things: A Review / F. C. Andriulo et al. *Informatics*. 2024. Vol. 11, no. 4. P. 71-82.

23. Security-aware Data-driven Intelligent Transportation Systems / J. Malik et al. *IEEE Sensors Journal*. 2020. P. 127-139. URL: https://doi.org/10.1109/jsen.2020.3012046 (date of access: 29.07.2025).

24. Researcher. Blockchain Technology: Advancing Data Integrity And Security In Modern Systems. *International Journal of Research In Computer Applications and Information Technology (IJRCAIT)*. 2024. Vol. 7, no. 2. P. 1574-1596.

25. Baisholan N., Kubayev K., Baisholan T. Modern tools for information security systems. *Physico-mathematical series*. 2021. Vol. 335, no. 1. P. 14-18.

26. Goli V. R. React native evolution, native modules, and best practices. *International journal of computer engineering and technology*. 2021. Vol. 12, no. 2. P. 73-85.

27. Goli V. R. Cross-Platform Mobile Development: Comparing React Native and Flutter, and Accessibility in React Native. *International Journal of Innovative Research in Computer and Communication Engineering*. 2023. Vol. 11, no. 03. P. 115-123.