Міністерство освіти і науки України

Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра комп'ютерної інженерії та охоронних систем

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

ХМАРНА ІНТЕЛЕКТУАЛЬНА ІНФОРМАЦІЙНА СИСТЕМА МОНІТОРИНГУ БЕЗПЕКИ ОБ'ЄКТА З ВІДЕОАНАЛІТИКОЮ ТА ПРОГНОЗУВАННЯМ РИЗИКІВ

CLOUD-BASED INTELLIGENT INFORMATION SYSTEM FOR FACILITY SECURITY MONITORING WITH VIDEO ANALYTICS AND RISK PREDICTION

спеціальність 126 Інформаційні системи та технології
(шифр і назва спеціальності)

освітня програма «Інформаційні системи та технології охорони і безпеки»
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи ІСТО-41
КОВШ Юрій Григорович
_____
(підпис)

Керівник:
к.т.н., доцент
ТЕРЛЕЦЬКИЙ Тарас Володимирович
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«___» _____ 2026 р.
Гарант освітньої програми:
к.т.н., доцент
ТЕРЛЕЦЬКИЙ Тарас Володимирович
_____
(підпис)

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет: *комп'ютерних та інформаційних технологій*
Кафедра: *комп'ютерної інженерії та безпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *126 Інформаційні системи та технології*
Освітня програма: «*Інформаційні системи та технології охорони і безпеки*»

ЗАТВЕРДЖУЮ
Завідувач кафедри КІБ
к.т.н., доцент Терлецький Т. В.
_____
«___» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

КОВШУ Юрію Григоровичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: *Хмарна інтелектуальна інформаційна система моніторингу безпеки об'єкта з відеоаналітикою та прогнозуванням ризиків (комплексна робота з Мальчевський А. В.).*

Керівник роботи: *к.т.н., доцент Терлецький Тарас Володимирович*
затверджені наказом закладу вищої освіти від «16» грудня 2026 р. № 529/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи: «30» травня 2026 р.

3. Вихідні дані до роботи: *Джерелом розробки є наукові праці та публікації у сфері систем відеомоніторингу, технічна документація використовуваних програмних засобів і мережевих протоколів, сучасні інтернет-ресурси технічного спрямування*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *Анотація. Вступ. Розділ 1. Аналітичний огляд стану предметної області (аналіз предмет-предметної області; огляд веб-орієнтованих систем; аналіз методів візуалізації; формулювання вимог та завдань). Розділ 2. Обґрунтування вибору засобів та методів реалізації (вибір технологій розробки; обґрунтування архітектури; засоби відображення відеопотоку інтеграція із серверною частиною). Розділ 3. Практична реалізація (розробка інтерфейсу; реалізація модулів відображення відео та подій; тестування системи). Загальні висновки та рекомендації. Список використаних джерел. Додатки.*

5. Перелік графічного (ілюстративного) матеріалу: *Презентація на 15 слайдах*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Розділ 1 Аналітичний огляд стану предметної області	*Терлецький Т. В.*		
Розділ 2 Обґрунтування вибору засобів та методів реалізації	*Терлецький Т. В.*		
Розділ 3 Практична реалізація	*Терлецький Т. В.*		
Загальні висновки та рекомендації	*Терлецький Т. В.*		
Нормоконтроль	*Кайдик О. Л.*		
Гарант ОП	*Терлецький Т. В.*		
Показник запозичень тексту			
Академічна доброчесність	*Кайдик О. Л.*		

7. Дата видачі завдання: «16» грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1.	Обґрунтування теми	До 12.12.2025 р.	
2.	Огляд літератури із досліджуваної проблеми	До 12.12.2025 р.	
3.	Розділ 1 Аналітичний огляд стану предметної області	До 28.02.2026 р.	
4.	Розділ 2 Обґрунтування вибору засобів та методів реалізації	До 31.03.2026 р.	
5	Розділ 3 Практична реалізація	До 30.04.2026 р.	
6.	Загальні висновки та рекомендації	До 16.05.2026 р.	
7.	Формування списку використаних джерел	До 20.05.2026 р.	
8.	Формування додатків.	До 20.05.2026 р.	
9.	Формування презентації за темою кваліфікаційної роботи	До 20.05.2026 р.	
10.	Нормоконтроль	До 21.05.2026 р.	
11.	Інструментальна перевірка на академічний плагіат	До 22.05.2026 р.	
12.	Представлення кваліфікаційної роботи бакалавра до захисту	До 02.06.2026 р.	

Здобувач вищої освіти _____ (*Ковш Ю. Г.*)
(підпис)

Керівник кваліфікаційної роботи _____ (*Терлецький Т. В.*)
(підпис)

АНОТАЦІЯ

Ковш Ю. Г. Хмарна інтелектуальна інформаційна система моніторингу безпеки об’єкта з відеоаналітикою та прогнозуванням ризиків. Рукопис.

Кваліфікаційна робота бакалавра ОП «Інформаційні системи та технології охорони і безпеки» спеціальності 126 Інформаційні системи та технології. Луцький національний технічний університет. Луцьк, 2026.

Кваліфікаційна робота складається з вступу, трьох розділів, загальних висновків та рекомендацій, списку використаних джерел, додатків.

У першому розділі проаналізовано предметну область систем відеомоніторингу. Обґрунтовано актуальність розробки, сформульовано вимоги до систем реального часу та розглянуто існуючі рішення з точки зору архітектури і протоколів передачі відео.

У другому розділі обґрунтовано вибір технологічного стеку та архітектури клієнтської частини застосунку. Проведено порівняльний аналіз сучасних веб-фреймворків і протоколів потокової трансляції для забезпечення ефективної обробки відеоданих у режимі реального часу.

У третьому розділі описано процес реалізації клієнтської частини системи відеомоніторингу. Розглянуто механізми відображення потокового відео, візуалізації результатів відеоаналітики, організації журналу подій та побудови аналітичних дашбордів. Проведено тестування основних функціональних модулів застосунку.

Ключові слова: інформаційна система, відеомоніторинг, клієнтський застосунок, архітектура системи, обробка даних, реальний час, інтерфейс користувача, безпека.

ANNOTATION

Kovsh Y. Cloud-based intelligent information system for facility security monitoring with video analytics and risk prediction. Manuscript.

Bachelor's thesis in Information Systems and Technologies for Security and Safety, specialty 126 Information Systems and Technologies. Lutsk National Technical University. Lutsk, 2026.

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

The first chapter analyzes the subject area of video surveillance systems. It justifies the relevance of the development, formulates requirements for real-time systems, and examines existing solutions in terms of architecture and video transmission protocols.

The second chapter justifies the choice of the technology stack and the architecture of the application's client-side. A comparative analysis of modern web frameworks and streaming protocols is conducted to ensure efficient real-time video data processing.

The third chapter describes the process of implementing the client-side component of the video surveillance system. It examines mechanisms for displaying live video streams, visualizing video analytics results, organizing event logs, and building analytical dashboards. Basic functional modules of the application have been tested.

Keywords: information system, video monitoring, client application, system architecture, data processing, real time, user interface, security.

ЗМІСТ

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій та зростання вимог до безпеки об’єктів різного призначення особливої актуальності набувають системи відеомоніторингу. Вони забезпечують контроль ситуації в режимі реального часу, дозволяють своєчасно виявляти потенційні загрози та підтримують процес прийняття рішень оператором. Водночас традиційні підходи до побудови таких систем не завжди відповідають сучасним вимогам щодо продуктивності, масштабованості та інтерактивності.

Значною проблемою є ефективне відображення відеопотоків разом із результатами відеоаналітики в межах єдиного користувацького інтерфейсу. Збільшення кількості джерел відеоданих та необхідність їх одночасної обробки вимагають використання сучасних технологічних рішень, що забезпечують мінімальні затримки та високу швидкість реагування. У зв’язку з цим актуальним є розроблення клієнтської частини інформаційної системи відеомоніторингу, здатної ефективно працювати в умовах реального часу.

Метою кваліфікаційної роботи є розробка клієнтської частини інформаційної системи відеомоніторингу, що забезпечує відображення відеопотоків, обробку подій у режимі реального часу та підтримку ситуаційної обізнаності користувача.

Об’єктом проектування є процес відеомоніторингу та забезпечення безпеки об’єктів із використанням інформаційних систем реального часу. Предметом проектування є методи, програмні засоби та архітектурні рішення розробки клієнтської частини інформаційної системи відеомоніторингу.

Предмет дослідження – методи, технології та архітектурні рішення реалізації клієнтської частини системи відеомоніторингу з підтримкою потокового відео, відеоаналітики та взаємодії в режимі реального часу.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД СТАНУ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Системи моніторингу безпеки з точки зору користувача

Системи моніторингу безпеки об’єктів є важливою складовою сучасної інфраструктури установ, підприємств та приватних об’єктів. Їх основним призначенням є забезпечення постійного спостереження за контрольованою територією, своєчасне виявлення потенційно небезпечних ситуацій та інформування відповідальних осіб про виникнення подій безпеки. У сучасних умовах зростання кількості загроз, підвищення вимог до оперативності реагування та розвитку цифрових технологій значна увага приділяється саме ефективності та зручності взаємодії користувача з такими системами.

З точки зору користувача, система моніторингу безпеки повинна забезпечувати швидкий доступ до актуальної інформації, наочне відображення подій та мінімізацію часу на прийняття рішень. Основними категоріями користувачів у цих системах виступають оператори та адміністратори. Оператор безпеки здійснює безпосередній моніторинг відеопотоків, аналіз подій та приймає рішення щодо реагування на інциденти. Адміністратор системи відповідає за налаштування параметрів роботи, керування користувачами, конфігурацію зон контролю та порогових значень ризику.

Для оператора ключовими вимогами є:

– безперервне відображення відеопотоку в режимі реального часу;

– автоматичне виділення потенційно небезпечних об’єктів або ситуацій;

– зручна навігація між камерами;

– швидкий доступ до журналу подій;

– чітке інформування про рівень ризику.

Адміністратор, у свою чергу, взаємодіє із системою з метою її конфігурації та підтримки працездатності. Для нього важливими є інструменти управління користувачами, налаштування прав доступу, визначення контрольованих зон та

параметрів оцінювання ризиків. Інтерфейс адміністратора має бути структурованим та забезпечувати безпечне виконання конфігураційних змін.

Традиційні системи відеоспостереження здебільшого орієнтовані на пасивний перегляд відеопотоку. У таких рішеннях основна відповідальність за виявлення інцидентів покладається на оператора. Проте зі зростанням кількості камер і обсягів відеоданих цей підхід стає неефективним. Людина фізично не здатна постійно підтримувати високий рівень концентрації, що призводить до пропуску важливих подій.

У процесі роботи оператор часто перебуває в умовах підвищеного когнітивного навантаження, оскільки повинен одночасно контролювати декілька джерел відеоінформації. Тому система має зменшувати кількість ручних дій та автоматизувати рутинні процеси, зокрема виявлення об’єктів, підрахунок кількості осіб у зоні та формування подій безпеки.

Ефективність роботи оператора системи моніторингу безпеки безпосередньо пов’язана з рівнем його ситуаційної обізнаності. Під цим поняттям розуміють здатність людини сприймати інформацію про стан середовища, правильно її розуміти та прогнозувати подальший розвиток подій. У контексті систем відеомоніторингу ситуаційна обізнаність передбачає своєчасне виявлення змін у контрольованій зоні, розуміння характеру події та прийняття відповідного рішення щодо реагування [1].

Недостатній рівень ситуаційної обізнаності може призвести до запізнілого реагування або неправильного трактування інциденту. Тому інтерфейс системи повинен сприяти швидкому сприйняттю інформації, мінімізувати кількість відволікаючих елементів та акцентувати увагу користувача на критично важливих подіях.

Сучасні тенденції розвитку систем моніторингу безпеки передбачають інтеграцію методів комп’ютерного зору та автоматичного аналізу відеоданих [2]. Використання алгоритмів виявлення об’єктів дозволяє автоматично визначати присутність людей у кадрі, контролювати перебування в заборонених зонах та фіксувати перевищення допустимої кількості осіб. Для користувача це означає

перехід від пасивного спостереження до активного отримання структурованої інформації у вигляді подій безпеки.

Значну роль у підвищенні ефективності системи відіграє спосіб подання інформації. Візуалізація результатів відеоаналізу у вигляді графічних позначень, наприклад виділення об’єктів рамками, дозволяє оператору швидко ідентифікувати потенційно небезпечні ситуації. Додаткове відображення рівня ризику у вигляді текстових або графічних індикаторів сприяє оперативному прийняттю рішень.

Окремої уваги потребує питання доступності системи. У сучасних умовах користувачі очікують можливості роботи з будь-якого місця через мережу Інтернет без необхідності встановлення спеціалізованого програмного забезпечення. Це призводить до переходу від локальних систем до веб-орієнтованих рішень.

Таким чином, аналіз систем моніторингу безпеки з точки зору користувача показує, що ефективність їх функціонування значною мірою залежить не лише від технічних характеристик обладнання або алгоритмів обробки відео, а й від якості реалізації клієнтської частини. Користувацький інтерфейс визначає швидкість сприйняття інформації, рівень когнітивного навантаження, забезпечує формування ситуаційної обізнаності оператора та безпосередньо впливає на швидкість прийняття рішень. У зв’язку з цим розробка сучасної веб-орієнтованої клієнтської частини системи моніторингу безпеки є актуальним та практично важливим завданням.

1.2 Веб-орієнтовані інформаційні системи у сфері безпеки

Розвиток інформаційних технологій сприяв трансформації підходів до побудови систем моніторингу безпеки. Якщо раніше такі системи функціонували переважно як локальні програмно-апаратні комплекси з прив’язкою до конкретного робочого місця, то сьогодні все більшого поширення

набувають веб-орієнтовані інформаційні системи, що забезпечують доступ до функціоналу через мережу Інтернет.

Веб-орієнтована інформаційна система – це програмна система, доступ до якої здійснюється за допомогою веб-браузера або спеціалізованої десктопної оболонки на базі веб-технологій, а обробка даних розподілена між клієнтською та серверною частинами. Такий підхід дозволяє забезпечити незалежність від операційної системи користувача, спростити розгортання та оновлення програмного забезпечення, а також централізувати управління даними.

У сфері безпеки перехід до веб-орієнтованих рішень викликаний рядом факторів:

– необхідністю віддаленого доступу до відеопотоків та журналів подій;

– потребою централізованого зберігання даних;

– інтеграцією з іншими інформаційними системами підприємства;

– масштабованістю при збільшенні кількості камер та користувачів;

– впровадженням моделі VSaaS, що дозволяє використовувати хмарні ресурси для обробки відео замість дорогого локального обладнання [3].

Традиційні настільні системи відеоспостереження характеризуються встановленням спеціалізованого програмного забезпечення на конкретному комп’ютері. Оновлення таких систем потребує ручного втручання, а доступ до даних часто обмежений межами локальної мережі. Крім того, масштабування подібних рішень може вимагати значних апаратних ресурсів на кожному робочому місці.

З іншого боку, веб-орієнтовані системи дозволяють користувачам працювати з будь-якого пристрою, що має доступ до мережі та сучасний браузер. Це особливо важливо для керівників або відповідальних осіб, які потребують оперативного доступу до інформації незалежно від місця перебування. Таким чином, веб-підхід підвищує мобільність та гнучкість управління безпекою об’єкта.

Важливою складовою сучасних веб-систем є використання хмарних технологій. Хмарна інфраструктура забезпечує централізоване зберігання даних,

резервування інформації та можливість динамічного масштабування ресурсів залежно від навантаження. У системах відеомоніторингу це дозволяє ефективно обробляти великі обсяги відеоданих та подій безпеки без необхідності розгортання складної локальної інфраструктури.

Архітектурно веб-орієнтовані системи безпеки зазвичай реалізуються на основі моделі «клієнт–сервер». Серверна частина відповідає за зберігання даних, обробку запитів, реалізацію бізнес-логіки та взаємодію з модулями відеоаналітики. Клієнтська частина забезпечує відображення інформації, взаємодію з користувачем та передачу запитів до сервера. Взаємодія між ними здійснюється через програмний інтерфейс прикладного програмування REST API (Додаток А) та протоколи реального часу, що забезпечують миттєву доставку сповіщень про події безпеки на екран оператора.

Особливістю веб-орієнтованих систем у сфері безпеки є необхідність забезпечення високого рівня надійності та захищеності. Оскільки передача даних здійснюється через відкриті мережі, обов’язковим є використання протоколів шифрування, механізмів автентифікації та контролю доступу [4]. Крім того, потрібно забезпечити стабільність відображення відеопотоку та коректність передачі подій у режимі реального часу.

З точки зору користувача веб-орієнтована система повинна забезпечувати:

– швидке завантаження інтерфейсу;

– оновлення даних без перезавантаження сторінки;

– адаптивність до різних розмірів екранів;

– інтерактивність та зручність навігації;

– можливість одночасного відображення декількох інформаційних блоків.

Сучасні тенденції розвитку веб-технологій сприяють створенню інтерактивних односторінкових застосунків [5]. У таких застосунках основна логіка відображення реалізується на стороні клієнта за допомогою сучасних JavaScript-фреймворків. Це дозволяє уникнути повторного завантаження елементів інтерфейсу, забезпечує плавність переходів та створює умови для підтримки високого рівня ситуаційної обізнаності користувача.

Незважаючи на очевидні переваги масштабованості та віддаленого доступу, ефективність веб-орієнтованих систем цілком залежить від стабільності фронтенд-архітектури. Вона має гарантувати безперервне відображення важких відеоданих та надійну інтеграцію із серверними компонентами.

1.3 Методи візуалізації даних та подій безпеки

У сучасних інформаційних системах, орієнтованих на обробку подій інформаційної безпеки, візуалізація даних є одним із ключових елементів ефективного функціонування системи. Значні обсяги подій, що генеруються різними мережевими пристроями, серверами, користувацькими діями, потребують не лише зберігання та аналізу, але й візуалізації для забезпечення оперативного реагування. Без відповідних методів візуалізації навіть правильно зібрані та оброблені дані можуть бути складними для розуміння.

Методи візуалізації даних у системах моніторингу безпеки базуються на перетворенні числової, текстової та часової інформації у графічні або структуровані форми представлення. До таких методів належать табличне відображення журналів подій, графічне подання статистики інцидентів, використання діаграм для демонстрації динаміки змін, а також інтерактивні панелі моніторингу, що відображають поточний стан системи в реальному часі. Велике значення має відображення часових залежностей, оскільки більшість інцидентів інформаційної безпеки аналізуються саме в контексті їх виникнення та розвитку у часі.

Одним із найбільш ефективних методів представлення результатів інтелектуального аналізу є використання графічних накладень безпосередньо поверх відеопотоку [6]. Важливу роль при цьому відіграє динамічне кольорове кодування: використання різних кольорів залежно від стану об’єкта дозволяє на підсвідомому рівні класифікувати події за ступенем важливості. Це значно знижує когнітивне навантаження, оскільки користувачу не потрібно самостійно виявляти рух на фоні статичних об’єктів.

Для аналізу подій у часовому вимірі використовуються методи журналізації та візуалізації послідовностей. Журнал подій безпеки, представлений у вигляді структурованого списку з візуальними маркерами та стоп-кадрами, дозволяє швидко переходити до критичних фрагментів архіву. Водночас для оцінювання загального стану безпеки об’єкта доцільно застосовувати методи статистичної візуалізації у формі інтерактивних дашбордів. Використання лінійних графіків для відображення динаміки рівня ризику та діаграм для ілюстрації розподілу подій за категоріями дозволяє адміністратору системи виявляти закономірності та прогнозувати потенційні загрози на основі накопичених даних [7].

При виборі методів візуалізації для веб-орієнтованих систем необхідно враховувати також ергономічні вимоги до інтерфейсів. Використання темних тем оформлення є стандартом для професійних систем моніторингу, оскільки це зменшує зорову втому та дозволяє акцентувати увагу на яскравих індикаторах подій [8]. Принцип мінімалізму при проектуванні панелей керування забезпечує відсутність відволікаючих факторів, що є дуже важливим під час обробки інцидентів у стресових умовах.

Правильно підібрані методи візуалізації не просто покращують естетику інтерфейсу, а й безпосередньо впливають на швидкість ухвалення рішень. Вони дозволяють оператору миттєво класифікувати загрози та знижують ризик пропуску критичного інциденту через втому.

1.4 Аналіз користувацьких інтерфейсів сучасних систем відеомоніторингу

Проектування клієнтської частини інтелектуальної системи потребує ретельного вивчення існуючих рішень на ринку для виявлення найбільш ефективних практик та типових недоліків. Сучасні системи відеомоніторингу можна розділити на дві великі групи: професійні локальні комплекси, такі як Milestone XProtect чи Hikvision iVMS, та хмарні веб-орієнтовані платформи, до

яких належать Verkada, Ajax Systems або хмарні модулі Dahua [9-11]. Аналіз цих рішень дозволяє сформувати перелік вимог до інтерфейсу проектованої системи.

Характерною особливістю сучасних інтерфейсів є наявність панелі керування подіями, яка відображає повідомлення про тривоги, детекцію руху, спроби несанкціонованого доступу або інші критичні ситуації. Інформація зазвичай структурується за рівнями пріоритетності та супроводжується кольоровими індикаторами, що дозволяє оператору швидко визначити ступінь загрози. Важливою складовою є інтеграція відео з аналітичними модулями, зокрема системами розпізнавання облич, номерних знаків або поведінкових аномалій.

Сучасні веб-орієнтовані системи безпеки сповідують принцип «Cloud-first» та орієнтуються на простоту використання. Такі системи як Verkada демонструють високий рівень інтеграції відеоаналітики в інтерфейс: результати розпізнавання об’єктів відображаються у вигляді лаконічних карток з метаданими, а пошук в архіві здійснюється за візуальними ознаками.

Окремо варто відзначити підхід до візуалізації мобільних та веб-інтерфейсів компанії Ajax Systems. Їхні рішення вирізняються високою швидкістю відгуку та використанням нативних елементів керування. Це забезпечує безперервність користувацького досвіду. Проте в таких системах акцент часто зміщується у бік охоронної сигналізації,. А от глибока відеоаналітика з виведенням складних графіків ризиків у реальному часі залишається другорядною функцією.

Аналіз існуючих рішень показує, що більшість сучасних інтерфейсів систем відеомоніторингу мають наступні спільні тенденції: перехід до мінімалістичного дизайну, використання інтерактивних мап об’єкта, ергономічність, інтуїтивність та можливість адаптації під потреби конкретного об’єкта.

Водночас аналіз показує і ряд типових проблем. У професійних комплексах інтерфейс часто перевантажений великою кількістю функціональних елементів, що ускладнює навчання нових користувачів та

підвищує когнітивне навантаження на оператора. У деяких веб-орієнтованих рішеннях відеопотік, журнал подій та аналітичні графіки розміщуються у різних вкладках або розділах. Це знижує цілісність сприйняття інформації та негативно впливає на рівень ситуаційної обізнаності. Крім того, частина хмарних сервісів обмежує глибину налаштування аналітики або не надає розширених інструментів статистичного аналізу ризиків.

Під час розробки клієнтської частини доцільно поєднати простоту сучасних хмарних веб-платформ із глибиною налаштувань локальних комплексів. Створення єдиного робочого простору, де відео, зони та журнал інтегровані в межах одного екрана, є ключем до мінімізації когнітивного навантаження персоналу.

1.5 Постановка завдань на кваліфікаційну роботу бакалавра

На основі проведеного аналізу систем моніторингу безпеки з точки зору користувача, дослідження веб-орієнтованих інформаційних систем та огляду сучасних інтерфейсів відеомоніторингу можна сформулювати сукупність вимог до проектованої інформаційної системи. Визначення вимог є ключовим етапом проектування, оскільки саме вони задають функціональні межі системи, визначають її архітектурні особливості та критерії оцінювання ефективності.

Насамперед необхідно забезпечувати безперервний моніторинг відеопотоку в режимі реального часу з можливістю відображення результатів автоматичного аналізу кадрів. Виявлення об’єктів, підрахунок кількості осіб у контрольованій зоні та формування подій безпеки мають здійснюватися автоматично з мінімальним залученням оператора. Користувач повинен отримувати структуровану інформацію у вигляді подій із зазначенням часу, типу інциденту та рівня ризику.

Функціональні вимоги до системи передбачають реалізацію механізмів реєстрації та автентифікації користувачів, впровадження рольової моделі доступу для оператора та адміністратора, ведення журналу подій та можливість

пошуку інформації за визначеними параметрами. Система має підтримувати збереження історії інцидентів та формування статистичних звітів щодо стану безпеки об’єкта.

Зважаючи на сучасні тенденції розвитку інформаційних технологій проектована система має бути реалізована як веб-орієнтований застосунок із клієнт-серверною архітектурою. Клієнтська частина повинна забезпечувати інтерактивне оновлення даних без повного перезавантаження сторінки, підтримку роботи в різних середовищах та можливість інтеграції в настільну оболонку на базі веб-технологій. Серверна частина повинна реалізовувати обробку запитів, збереження даних, взаємодію з модулями відеоаналітики та передачу подій у режимі реального часу.

Окрему групу становлять нефункціональні вимоги. Система повинна характеризуватися високою надійністю та стабільністю роботи, забезпечувати захист даних під час передачі через мережу, а також підтримувати механізми контролю доступу та реєстрацію подій у системних журналах. Важливою вимогою є масштабованість, що дозволить розширювати систему при збільшенні кількості камер, користувачів або обсягів оброблюваних даних без суттєвої зміни архітектури.

Значна увага має бути приділена вимогам до користувацького інтерфейсу. Інтерфейс повинен забезпечувати формування єдиного робочого простору, в межах якого одночасно відображаються відеопотік, результати аналітики, журнал подій та індикатори рівня ризику. Дизайн має бути мінімалістичним, з акцентом на критично важливих елементах, що сприятиме зниженню когнітивного навантаження та підтримці високого рівня ситуаційної обізнаності оператора. Важливо забезпечити інтуїтивну навігацію, швидкий доступ до ключових функцій та адаптивність інтерфейсу до різних розмірів екранів.

Для реалізації поставленої мети в межах кваліфікаційної роботи необхідно вирішити наступні завдання:

1) обґрунтувати вибір технологічного стеку (фреймворків, бібліотек та середовища виконання) для розробки клієнтської частини системи;

2) спроектувати архітектуру фронтенд-застосунку, забезпечивши ефективну взаємодію з хмарним сервером та модулями штучного інтелекту;

3) розробити дизайн користувацького інтерфейсу для ролей оператора та адміністратора, орієнтований на підтримку ситуаційної обізнаності;

4) реалізувати модуль відображення відеопотоку в режимі реального часу з підтримкою динамічного накладення результатів відеоаналітики;

5) розробити інтерактивні аналітичні панелі для візуалізації рівнів ризику та фіксації історії подій безпеки;

6) інтегрувати клієнтську частину з Backend-API за допомогою протоколів передачі даних у реальному часі;

7) провести тестування розробленого програмного забезпечення на відповідність вимогам продуктивності та зручності використання.

Виконання поставлених завдань дозволить створити сучасну інформаційну систему, яка не лише фіксує події, а й допомагає користувачу ефективно прогнозувати ризики та оперативно реагувати на загрози безпеці об’єкта.

РОЗДІЛ 2
ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ ТА МЕТОДІВ РЕАЛІЗАЦІЇ

2.1 Вибір технологій розробки клієнтської частини веб-застосунку

Розробка клієнтської частини сучасної веб-орієнтованої інформаційної системи потребує обґрунтованого вибору технологій, що забезпечують високу продуктивність, масштабованість та зручність взаємодії користувача з системою. Враховуючи вимоги, сформульовані у першому розділі, зокрема необхідність обробки даних у режимі реального часу, відображення відеопотоку, інтеграції результатів відеоаналітики та підтримки ситуаційної обізнаності користувача, вибір технологічного стеку має базуватися на принципах інтерактивності, гнучкості та розширюваності.

Одним із ключових рішень при проектуванні клієнтської частини є вибір архітектурної моделі застосунку. У сучасних умовах найбільш доцільним підходом є використання моделі односторінкового застосунку. Вона дозволяє забезпечити динамічне оновлення інтерфейсу без повного перезавантаження сторінок. Такий підхід є особливо важливим для систем відеомоніторингу, де безперервність відображення інформації відіграє критичну роль.

Для реалізації цієї моделі обрано бібліотеку React [12]. Її вибір пояснюється здатністю швидко оновлювати інтерфейс за допомогою віртуального DOM, а також зручністю розробки окремих компонентів. Це дозволяє скоротити кількість прямих звернень до реального DOM і забезпечити високу продуктивність при частих оновленнях інтерфейсу, характерних для систем відеомоніторингу. Додатковою перевагою є можливість ізолювати логіку кожного елемента екрану. Можна незалежно оновлювати окремі елементи, наприклад, панель тривог або конкретний відеоплеєр, не сповільнюючи роботу всього застосунку.Порівняння React з іншими популярними інструментами наведено у таблиці 2.1.

Для організації централізованого управління станом застосунку обрано Redux Toolkit (RTK). Специфіка систем відеомоніторингу полягає в одночасній

обробці великих масивів даних різних типів: статусів камер, поточних подій безпеки, аналітичних показників та конфігурації самого інтерфейсу. Використання єдиного сховища дозволяє забезпечити узгодженість цих даних та уникнути ситуацій, коли різні частини інтерфейсу відображають різну інформацію. Крім того, використання Redux Toolkit спрощує реалізацію логіки обробки подій та забезпечує передбачуваність змін стану [13].

Таблиця 2.1 – Порівняння JavaScript-фреймворків для систем реального часу

Критерій порівняння	React	Angular	Vue.js
Архітектурний підхід	Бібліотека (висока гнучкість)	Повноцінний фреймворк (жорстка структура)	Прогресивний фреймворк
Механізм оновлення	Virtual DOM (висока швидкість)	Change Detection (складніший при великих даних)	Virtual DOM / Reactivity
Продуктивність	Висока	Стабільна, залежить від оптимізації	Висока
Складність освоєння	Середня (велика спільнота)	Висока (потребує знання RxJS)	Низька / Середня
Управління станом	Зовнішні бібліотеки (Redux, Zustand, Context, тощо)	Вбудовані сервіси на базі RxJS	Спеціалізовані рішення (Pinia, Vuex)
Робота з відеопотоками	Добре підходить завдяки ефективному оновленню компонентів	Потребує ручної оптимізації при великій кількості потоків	Висока ефективність завдяки точному відстеженню реактивних залежностей

Оскільки система буде виконувати роль робочого місця оператора, доцільним є використання десктопного середовища виконання. Для цього обрано Electron, який дозволяє створювати настільні застосунки на основі веб-технологій [14]. Він дозволяє поєднати гнучкість сучасних веб-технологій із прямим доступом до файлової системи та апаратних ресурсів комп’ютера, що неможливо реалізувати у звичайному браузері.

Для організації передачі відеопотоків у системі обрано сервер потокового відео MediaMTX. Він діє як проміжний шар між джерелами відео та клієнтською частиною. Його використання дозволяє забезпечити сумісність між різними

протоколами передачі даних та підготувати відеопотік до ефективного відображення у клієнтському застосунку [15].

Для мережевої взаємодії та передачі тривожних сповіщень у реальному часі оптимальним рішенням є протокол WebSockets, який забезпечує постійне двостороннє з’єднання між клієнтом і сервером. На відміну від традиційних регулярних запитів до сервера, такий підхід сильно зменшує затримки та знижує навантаження на мережу [16].

Як інструмент для збірки та оптимізації коду обрано Vite. Він використовує нативні ES-модулі, що забезпечує швидке оновлення окремих компонентів і прискорює етапи розробки та налагодження інтерфейсів [17]. Для стилізації компонентів надано перевагу препроцесору Sass (SCSS). Його інструментарій дозволяє впровадити системний підхід до дизайну через використання змінних та міксинів, забезпечуючи візуальну цілісність інтерфейсу [18].

Обраний технологічний стек формує цілісну основу для створення продуктивного та масштабованого клієнтського застосунку. Поєднання сучасних веб-технологій із можливостями десктопного середовища дозволяє забезпечити ефективну обробку відеопотоків, підтримку роботи в режимі реального часу та зручну взаємодію користувача із системою, що відповідає вимогам до сучасних систем відеомоніторингу.

2.2 Обґрунтування архітектури клієнтської частини

Ефективність функціонування клієнтської частини інформаційної системи значною мірою залежить від правильно обраної архітектури застосунку. Архітектура визначає спосіб організації компонентів, їх взаємодію, а також впливає на продуктивність, масштабованість і зручність супроводу програмного забезпечення. Враховуючи вимоги до застосунку, архітектура клієнтської частини повинна забезпечувати гнучкість, передбачуваність та ефективне управління станом.

В основі клієнтської частини доцільно використовувати компонентно-орієнтований підхід, характерний для сучасних веб-застосунків. Такий підхід передбачає розбиття інтерфейсу на окремі, логічно завершені елементи – компоненти, кожен з яких відповідає за окрему функціональну частину системи. Така організація дозволяє забезпечити повторне використання коду, спростити тестування та підвищити зручність супроводу системи. Крім того, компонентна модель сприяє локалізації змін: модифікація окремого елемента інтерфейсу не впливає на інші частини системи. Найбільш важливим це є при розширенні функціональності.

Враховуючи використання середовища Electron, архітектурне рішення базується на багатопроцесній моделі, що включає головний процес та процес рендерингу. Головний процес відповідає за взаємодію з операційною системою, управління життєвим циклом застосунку та доступ до системних ресурсів. Процес рендерингу забезпечує відображення інтерфейсу користувача та виконує клієнтську логіку, реалізовану з використанням бібліотеки React.

Поділ на окремі процеси дозволяє ізолювати інтерфейс від складних фонових задач та прямої взаємодії з операційною системою. Це особливо важливо для систем відеомоніторингу, де одночасно можуть оброблятися декілька відеопотоків та великий обсяг подій. Така ізоляція підвищує надійність системи, оскільки збій в одному процесі не призводить до повного припинення роботи застосунку.

Обмін даними між процесами відбувається за допомогою вбудованого механізму міжпроцесної взаємодії. Це дозволяє безпечно передавати дані між головним процесом і користувацьким інтерфейсом, зокрема при роботі з відеопотоками, файлами або налаштуваннями системи, не порушуючи цілісності архітектури.

Загальну схему архітектури клієнтської частини та її взаємодію із зовнішніми серверними компонентами наведено на рисунку 2.1.

Представлена на рисунку 2.1 архітектура відображає запропоновану внутрішню структуру клієнтської частини та її зв’язок із зовнішніми модулями.

В основі візуального інтерфейсу лежить бібліотека React, яка на рівні процесу рендерингу відповідатиме за відображення відеопотоків, подій безпеки та аналітичної інформації. Логіку управління станом застосунку передбачається побудувати за допомогою централізованого сховища Redux, що реалізує концепцію єдиного джерела істини та забезпечує узгодженість даних між усіма компонентами. При цьому всі звернення до апаратних ресурсів комп'ютера або файлової системи делегуються головному процесу виключно через механізм міжпроцесної комунікації.

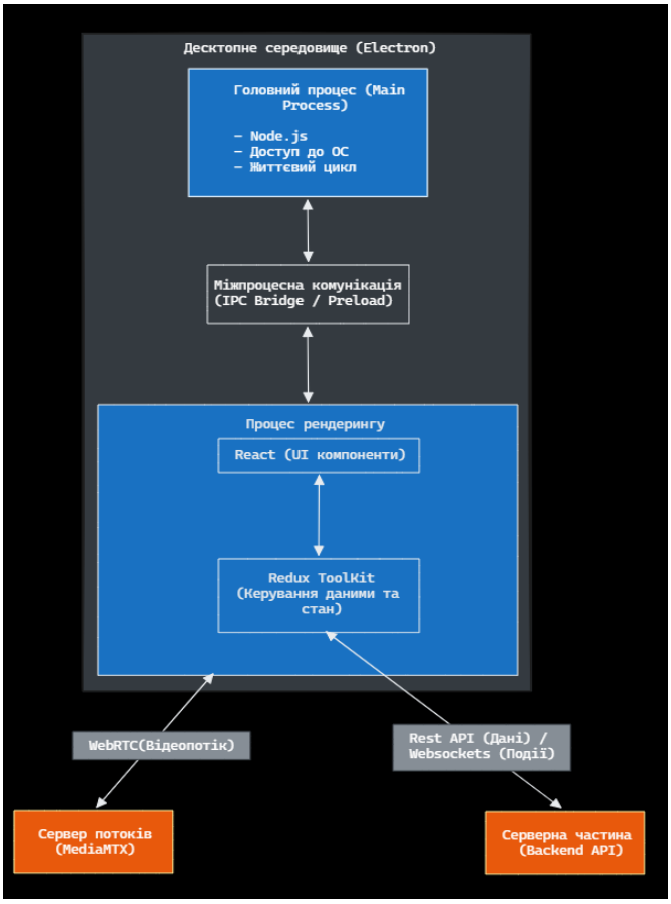


Рисунок 2.1 – Архітектура клієнтської частини застосунку

Взаємодія із серверною частиною реалізується через два основні канали. Для передачі структурованих даних, таких як інформація про камери, користувачів та налаштування, обрано архітектуру REST API. Для отримання подій безпеки в режимі реального часу оптимальним є використання WebSocket-

з’єднання, що забезпечує миттєву доставку інформації до клієнта без необхідності постійного виконування запитів до сервера.

Окремо в архітектурі виділяється підсистема передачі відео. Потоки з камер обробляються сервером потоків MediaMTX з подальшою доставкою до клієнтської частини через механізм потокової трансляції в режимі реального часу. Такий підхід дозволяє забезпечити мінімальну затримку відображення відео та стабільність роботи при змінних умовах мережі.

Використання запропонованого архітектурного підходу створює основу для побудови стабільної та гнучкої клієнтської частини, здатної ефективно функціонувати в умовах постійного надходження даних та забезпечувати подальший розвиток системи.

2.3 Обґрунтування засобів відображення відеопотоку

Одним із ключових компонентів системи відеомоніторингу є підсистема відображення відеопотоку, яка забезпечує безперервну передачу та візуалізацію відеоданих у режимі реального часу. Від ефективності її реалізації залежить якість сприйняття інформації користувачем, швидкість реагування на інциденти та загальний рівень ситуаційної обізнаності.

Основною вимогою до засобів відображення відео є мінімальна затримка між моментом отримання даних із джерела та їх візуалізацією в клієнтському інтерфейсі. У системах безпеки навіть незначні затримки можуть призводити до зниження ефективності реагування, тому обрані технології повинні забезпечувати передачу відео з мінімальними втратами часу.

У сучасних системах відеомоніторингу застосовуються різні протоколи передачі відео, які виконують різні функції в межах загальної архітектури. Зокрема, протоколи типу RTSP використовуються для отримання відеопотоку від джерел, тоді як технології на зразок WebRTC застосовуються для передачі відео до клієнтського застосунку [19, 20]. Для обґрунтування вибору підходу доцільно порівняти їх характеристики (табл. 2.2).

Сегментовані підходи, що базуються на протоколі TCP, передбачають розбиття відеопотоку на окремі фрагменти. Це забезпечує стабільність відтворення, але необхідність накопичення кількох сегментів у буфері клієнта створює значну затримку. Натомість технологія WebRTC, яка використовує протокол UDP, дозволяє передавати кадри миттєво. На відміну від TCP, UDP не чекає підтвердження отримання кожного пакету, що дозволяє уникнути ефекту «накопиченої затримки» при тимчасових перешкодах у мережі.

Таблиця 2.2 – Порівняльна характеристика протоколів передачі відео

Критерій порівняння	Сегментовані протоколи (HLS, DASH)	Потокові протоколи (RTSP)	Технології реального часу (WebRTC)
Транспортний протокол	TCP (HTTP)	TCP / UDP	Переважно UDP
Затримка	Висока (5-30 сек)	Низька (1-2 сек)	Ультранизька (< 500 мс)
Механізм передачі	Завантаження файлів-сегментів	Безперервний потік	Peer-to-Peer / Медіа-сервер
Нативна підтримка	Повна (через JS-плеєри)	Відсутня у браузерах	Повна (API браузера)
Стабільність зв’язку	Дуже висока	Середня	Висока (адаптивний бітрейт)

Оптимальним рішенням є використання зв’язки протоколу RTSP для отримання даних із камер та WebRTC для доставки відео в інтерфейс. Оскільки браузери не підтримують RTSP нативно, виникає необхідність застосування проміжного сервера потоків. У цьому випадку використовують MediaMTX, який виконує роль медіа-проксі та здійснює перепакування потоку без перекодування, що зберігає ресурси процесора та не вносить додаткових затримок.

У клієнтській частині планується формувати унікальні ідентифікатори відеопотоків на основі параметрів джерела для гнучкого підключення до сервера. Отримане URL-посилання ініціалізуватиме з’єднання та інтегруватиме відеодані у відповідний компонент відображення.

Раціональним кроком є розділення каналів передачі відеоданих та супровідної аналітичної інформації. Медіапотік транслюватиметься безперервно через WebRTC, тоді як результати аналітики, серед яких інформація про події та координати об’єктів, надходитимуть паралельно через механізм обміну даними

у реальному часі. Такий підхід суттєво зменшить мережеве навантаження. Водночас на рівні користувацького інтерфейсу буде реалізована синхронізація цих процесів: графічні елементи оновлюватимуться чітко відповідно до зміни стану відео, створюючи єдине інтерактивне середовище моніторингу.

Запропонована архітектура підсистеми відображення повністю задовольняє ключову вимогу систем безпеки – забезпечення низької затримки візуалізації. Відокремлення логіки передачі важких медіаданих від потоку легкої аналітичної інформації гарантує високу продуктивність клієнтського застосунку та стабільність інтерфейсу навіть під час пікових навантажень.

2.4 Обґрунтування методів інтеграції з серверною частиною

Ефективна взаємодія клієнтської та серверної частин є важливою умовою функціонування інформаційної системи відеомоніторингу. Вибір методів інтеграції безпосередньо впливає на швидкість обміну даними, узгодженість інформації та здатність системи працювати в режимі реального часу.

Врахувавши особливості предметної області доцільно використовувати комбінований підхід до організації взаємодії між клієнтом і сервером, що поєднує запит-відповідь модель та механізми постійного з’єднання. Це дозволяє оптимально розподілити різні типи даних залежно від їх природи та вимог до швидкості обробки.

Для обміну структурованими даними (налаштування зон спостереження, інформація про камери, конфігурація системи) використовують архітектурний стиль REST. Такий підхід базується на використанні стандартних HTTP-запитів і забезпечує простоту реалізації, масштабованість та сумісність із сучасними веб-технологіями. REST-інтерфейс дозволяє організувати чітку структуру кінцевих точок доступу і забезпечити передбачувану взаємодію між клієнтом і сервером.

Разом із цим, покладання виключно на REST-архітектуру є недостатнім для модулів, що потребують миттєвого оновлення даних. Регулярні запити до сервера призводять до додаткового навантаження на мережу та збільшення

затримок при отриманні нової інформації. У зв’язку з цим для передачі подій безпеки необхідно застосовувати технології реального часу.

Для оперативної доставки подій до клієнтського застосунку обґрунтованим кроком є використання протоколу WebSocket, який підтримує постійний двосторонній канал зв’язку між клієнтом і сервером. Це дозволяє серверній частині ініціювати передачу даних без необхідності запиту з боку клієнта, що значно скорочує затримки та підвищує ефективність обміну інформацією.

Використання WebSocket особливо актуальне для передачі подій відеоаналітики, таких як виявлення об’єктів, зміна рівня ризику або виникнення інцидентів у зонах спостереження. Оперативне надходження цих даних гарантує своєчасне відображення інформації та підтримує високу ефективність роботи оператора.

Окрему роль у системі відіграє передача відеопотоків, яка має специфічні вимоги до пропускної здатності та затримки. З огляду на це інтеграцію відеоданих доцільно здійснювати через спеціалізований сервер потоків, який виконує функції ретрансляції та адаптації медіаданих. Такий підхід дозволяє відокремити обробку відео від основної серверної логіки та забезпечити більш ефективне використання ресурсів системи.

Важливим аспектом інтеграції є забезпечення безпеки взаємодії. Оскільки архітектура системи передбачає розмежування прав доступу на рівні оператора та адміністратора, обмін даними вимагає обов’язкової перевірки автентичності. Оптимальним рішенням виступає застосування токенів авторизації, які додаються до заголовків HTTP-запитів та використовуються при ініціалізації WebSocket-з’єднання. Це захищає систему від несанкціонованого доступу та гарантує цілісність конфігураційних налаштувань.

Уніфікація формату повідомлень також відіграє значну роль у побудові стабільної клієнт-серверної взаємодії. Для серіалізації інформації як у REST-запитах, так і в повідомленнях реального часу обрано формат JSON. Він має

нативну підтримку в клієнтському застосунку, що спрощує обробку даних і зменшує навантаження при великій кількості подій.

Отже, клієнтський застосунок взаємодіє з серверною частиною через REST-інтерфейс для отримання та зміни конфігураційних даних, WebSocket-з’єднання для отримання подій у режимі реального часу, а також через сервер потоків для відтворення відео. Розподіл каналів передачі даних відповідно до їх призначення дозволяє досягти балансу між продуктивністю, масштабованістю та зручністю реалізації.

Додатково доцільно враховувати використання механізмів обробки помилок та повторного підключення. Вони підтримуватимуть стабільність взаємодії у разі нестабільності мережевого з’єднання. Це особливо важливо для систем безперервного моніторингу, де втрата зв’язку може призвести до втрати критично важливої інформації.

Запропонований підхід дозволяє не лише забезпечити стабільну роботу системи в режимі реального часу, але й створює основу для її подальшого розширення та адаптації до змінних умов експлуатації.

РОЗДІЛ 3
ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Загальна структура клієнтської частини системи

Клієнтська частина розробленої системи відеомоніторингу виконана у вигляді односторінкового додатку, побудованого на бібліотеці React у поєднанні з десктопним середовищем Electron. Такий підхід поєднує переваги веб-технологій, зокрема динамічне оновлення інтерфейсу, гнучкість та проста розробка, та переваги десктопних додатків, включаючи доступ до ресурсів операційної системи, керування вікнами та робота в багатовіконному режимі.

Архітектура клієнтського модуля побудована за принципом модульності з чітким розподілом відповідальності між компонентами. Вся візуальна логіка інтерфейсу виконується у процесі рендерингу, глобальне керування життєвим циклом програми та взаємодія з операційною системою покладені на головний процес Electron. Взаємодія між цими частинами організована за допомогою механізму міжпроцесної комунікації, що забезпечує безпечний обмін даними.

Файлова структура проекту сформована за функціональним підходом. Це означає, що файли групуються виключно за їхнім призначенням та роллю в системі. Така ієрархія робить навігацію кодовою базою інтуїтивно зрозумілою, сильно спрощує подальший супровід програмного коду та спрощує подальше розширення системи. Загальна структура клієнтської частини наведена у лістингу 3.1.

Наведена структура відображає поділ застосунку на окремі функціональні модулі, що відповідають за інтерфейс користувача, взаємодію із серверною частиною, управління станом та допоміжні утиліти. Такий підхід спрощує супровід коду та дозволяє масштабувати систему.

Директорія `electron` містить файли, що відповідають за функціонування десктопного середовища. Файл `main.js` реалізує головний процес застосунку, забезпечує створення та керування вікнами, а також обробку системних подій. Файл `preload.js` виконує роль проміжного шару між головним процесом і

клієнтською частиною, забезпечуючи контрольований доступ до API операційної системи.

Лістинг 3.1 – Структура клієнтської частини застосунку

```
├── electron
│   ├── main.js
│   └── preload.js
├── src
│   ├── components
│   │   └── TitleBar
│   ├── hooks
│   │   ├── useEventStream.js
│   │   ├── useKioskMode.js
│   │   └── useRole.js
│   ├── pages
│   │   ├── CamerasGrid
│   │   ├── Login
│   │   ├── Monitoring
│   │   ├── Settings
│   │   └── Analytics
│   ├── services
│   │   ├── auth
│   │   ├── camerasApi.js
│   │   └── zonesApi.js
│   ├── store
│   │   └── store.js
│   ├── styles
│   │   └── global.scss
│   ├── utils
│   │   └── windowStorage.js
│   ├── App.jsx
│   └── main.jsx
├── Dockerfile
├── README.md
├── package.json
├── script.js
└── vite.config.js
```

кінець лістингу 3.1

Основна логіка клієнтської частини зосереджена у директорії `src`. Папка `components` містить універсальні елементи інтерфейсу, які застосовуються в різних частинах системи. Директорія `pages` включає компоненти, що відповідають окремим сторінкам застосунку, таким як сторінка моніторингу, сітка камер, налаштування, автентифікація та аналітика. Кожна сторінка

побудована як окремий модуль, із власними стилями та допоміжними компонентами.

Директорія `services` реалізує взаємодію із серверною частиною. Вона містить набір API-клієнтів, побудованих за допомогою технології RTK Query, для обміну даними про камери, тривожні зони та перевірки автентичності користувачів. Директорія `store` містить конфігураційний файл централізованого сховища Redux. Він виконує роль єдиної точки збирання стану: об’єднує локальні дані авторизації з кешованими мережевими відповідями від API-клієнтів, а також інтегрує відповідні механізми `middleware` для автоматичної обробки асинхронних запитів.

Внутрішню структуру цього сховища поділено на незалежні функціональні сегменти – слайси. Наприклад, слайс автентифікації відповідає за безпечне зберігання токенів доступу та атрибутів поточного користувача. Для роботи з сервером використовується RTK Query, який не тільки отримує дані, але й кешує їх та автоматично оновлює при змінах. Це дозволяє швидше відображати актуальну інформацію в інтерфейсі та зменшує кількість зайвих запитів. Локальний стан компонентів використовується виключно для керування елементами інтерфейсу, такими як розкриття модальних вікон чи перемикання режимів перегляду, щоб не перевантажувати центральне сховище.

Допоміжні функції винесені у директорію `utils`, а глобальні стилі застосунку – у `styles`. Таке розділення запобігає дублюванню фрагментів коду та суттєво підвищує загальну читабельність проекту.

Маршрутизація клієнтського застосунку реалізована із використанням бібліотеки React Router і забезпечує навігацію між основними функціональними модулями системи. Доступ до більшості сторінок обмежений і потребує попередньої автентифікації користувача. Відкритим залишається лише початковий шлях «`/login`», який ініціалізує модуль авторизації.

Після успішної перевірки прав система автоматично перенаправляє користувача до робочого простору. Кореневий маршрут «/» відповідає за завантаження головного екрана із загальною сіткою підключених відеокамер.

Для детального аналізу конкретної камери та пов’язаних із нею подій передбачено динамічний шлях «`/monitoring/:cameraId`», куди підставляється унікальний ідентифікатор обраного пристрою.

Окрему увагу приділено обробці результатів роботи системи: для перегляду статистичних зведень впроваджено захищений маршрут «`/analytics`». Налаштування конфігураційних параметрів відеомоніторингу винесено на ізольований шлях «`/settings`».

Взаємодія між клієнтським застосунком та серверною інфраструктурою здійснюється через два паралельні канали. Для отримання та модифікації структурованих даних, таких як конфігурації камер, зон та користувачів, використовуються REST-запити. Для забезпечення швидкого реагування на події безпеки застосовується постійне WebSocket-з’єднання, логіка якого винесена у користувацький хук `useEventStream`. Отримання відеопотоків реалізовано безпосередньо від сервера потоків MediaMTX за протоколом WebRTC, що гарантує мінімальну затримку відтворення.

Особливістю розробленої клієнтської частини є підтримка роботи в режимі кількох незалежних вікон, що є важливим для організації робочого місця оператора з кількома моніторами. Для того щоб налаштування не змішувалися між вікнами, використовується механізм міжпроцесної комунікації в середовищі Electron. Дані щодо конфігурації сітки камер зберігаються у локальному сховищі браузера з прив’язкою до унікального ідентифікатора вікна. Це дозволяє оператору формувати індивідуальну розкладку відеопотоків у кожному вікні окремо. Фрагмент вихідного коду, що відповідає за ініціалізацію та конфігурацію багатовіконного середовища в Electron, наведено у додатку В.

Розмежування прав доступу до функціоналу системи реалізовано безпосередньо на рівні клієнтської частини за допомогою розробленого хука `useRole`, який отримує рівень доступу з глобального стану після успішної автентифікації. Реалізацію логіки визначення ролі поточного користувача показано в лістингу 3.2. Користувач з роллю «Адміністратор» отримує повний доступ до системи, включаючи додавання нових камер, налаштування зон

аналітики та управління правами інших користувачів. Користувач з роллю «Оператор» має доступ виключно на перегляд відеопотоків, моніторинг журналу подій та перегляд існуючих зон безпеки без можливості їх редагування.

Лістинг 3.2 – Визначення ролі поточного користувача

```
export function useRole() {
    const user = useSelector(state => state.auth?.user);
    const role = user?.role || null;
    return {
        role,
        isAdmin: role === "ADMIN",
        isOperator: role === "OPERATOR",
    };
}
```

кінець лістингу 3.2

Додатково реалізовано механізм синхронізації стану автентифікації між вікнами через подію `storage` браузера, що забезпечує одночасний вихід із системи при натисканні кнопки виходу в будь-якому з відкритих вікон.

Таким чином, розроблена загальна структура клієнтської частини застосунку забезпечує модульність, масштабованість, чіткий розподіл функціональних обов’язків між модулями, ефективне управління станом та підтримку роботи в режимі реального часу. Обраний підхід дозволяє створити гнучку систему, яка відповідає сучасним вимогам до систем відеомоніторингу та здатна до подальшого розвитку та адаптації відповідно до змін вимог.

3.2 Проектування користувацького інтерфейсу

Візуальна концепція клієнтської частини ґрунтується на принципах мінімалізму, швидкої ситуаційної обізнаності та суворого розмежування функціонала за ролями. Усі екрани застосунку оформлено в єдиній темній кольоровій схемі з використанням глибоких відтінків синього та чорного. Таке рішення зменшує навантаження на очі під час тривалої роботи та є типовим для систем спостереження [1, 4].

Точкою входу до системи є екран автентифікації, який наведений на рисунку 3.1. У центральній частині інтерфейсу розміщено форму входу, яка відображається на темному фоні з ледь помітною сіткою. Таке рішення створює відчуття глибини інтерфейсу, не перевантажуючи його зайвими візуальними елементами. Картка авторизації містить логотип системи, поля для введення облікових даних із відповідними іконками, а також основну кнопку входу. Розташування елементів підібрано таким чином, щоб користувач міг швидко зорієнтуватися та виконати вхід без зайвих дій.

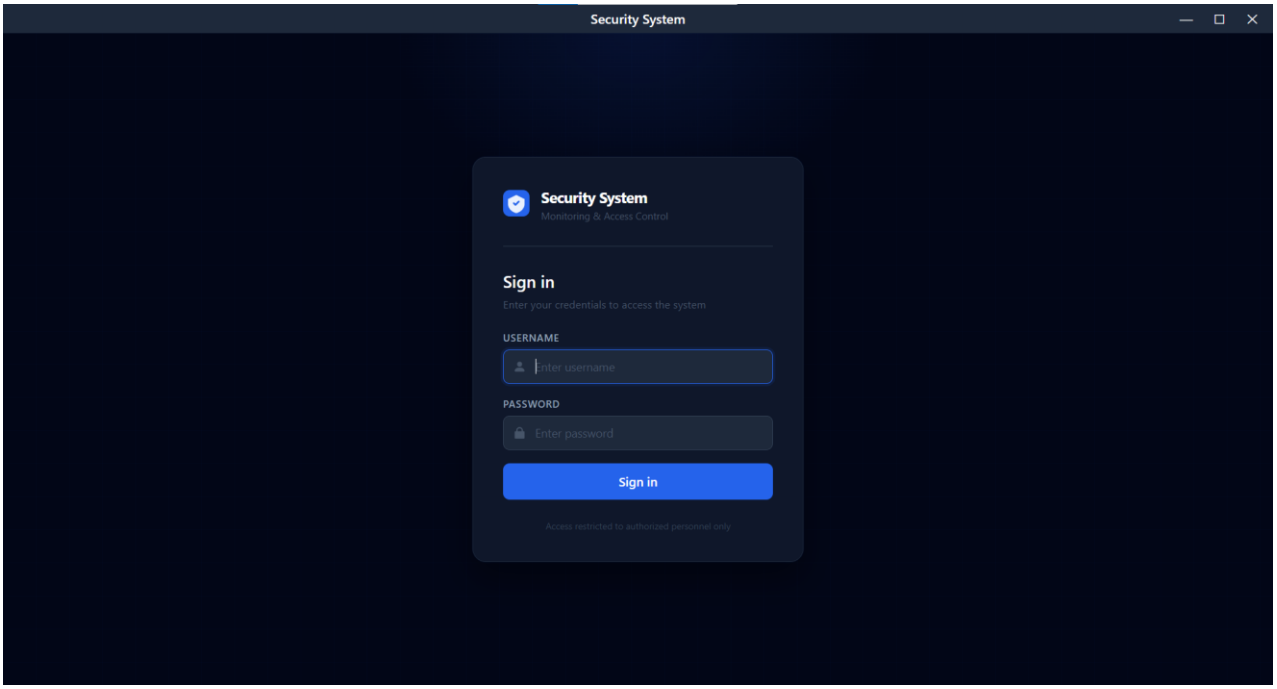


Рисунок 3.1 – Екран автентифікації користувача

Під час взаємодії з полями введення вони підсвічуються синім кольором, що дає користувачу зрозумілий візуальний відгук. У разі помилки введення інтерфейс відображає повідомлення з поясненням причини, що прискорює її виправлення. Після успішної автентифікації система обробляє отриманий токен, визначає рівень доступу користувача та автоматично перенаправляє його до головного екрану із сіткою камер.

Основний робочий простір, поданий на рисунку 3.2, реалізовано у вигляді адаптивної сітки відеопотоків. Користувач може самостійно змінювати кількість

відображуваних камер та їх розташування, зокрема використовувати сітки від чотирьох до шістнадцяти елементів. Вільні слоти позначаються пунктирним контуром із системним номером слота. При фіксації тривожної події відповідний відеопотік автоматично виділяється пульсуючою рамкою, колір якої показує рівень загрози (рис. 3.2).

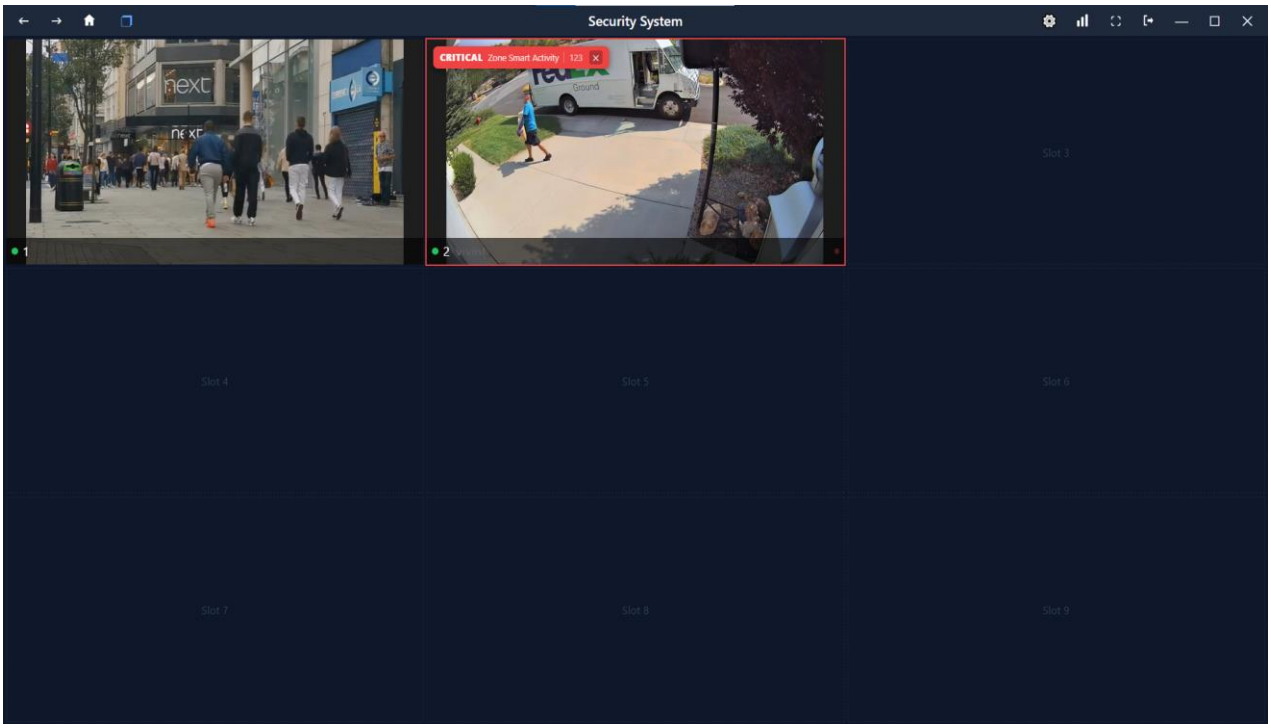


Рисунок 3.2 – Головний екран сітки камер

У кутку відеопотоку відображається додатковий інформаційний бейдж, який показує тип зафіксованого інциденту. Це сприяє швидкому розумінню характеру події та допомагає визначити проблемну зону при одночасному перегляді великої кількості камер.

У верхній частині розміщена навігаційна панель. Вона містить: стандартні кнопки керування вікном, кнопки навігації між сторінками, переходу до розділів налаштувань та аналітики, виходу з облікового запису, активації режиму моніторингу, а також можливість відкриття нового вікна застосунку, що є важливим для організації роботи на декількох моніторах. Наявність усіх основних елементів керування в одному місці дозволяє зменшити кількість зайвих дій та забезпечує швидкий доступ до ключових функцій системи.

Така організація робочого простору дозволяє оператору одночасно контролювати значну кількість відеопотоків без втрати концентрації. Використання візуальних індикаторів та структурованого розташування елементів інтерфейсу сприяє швидкому виявленню потенційно небезпечних ситуацій та оперативному реагуванню на них.

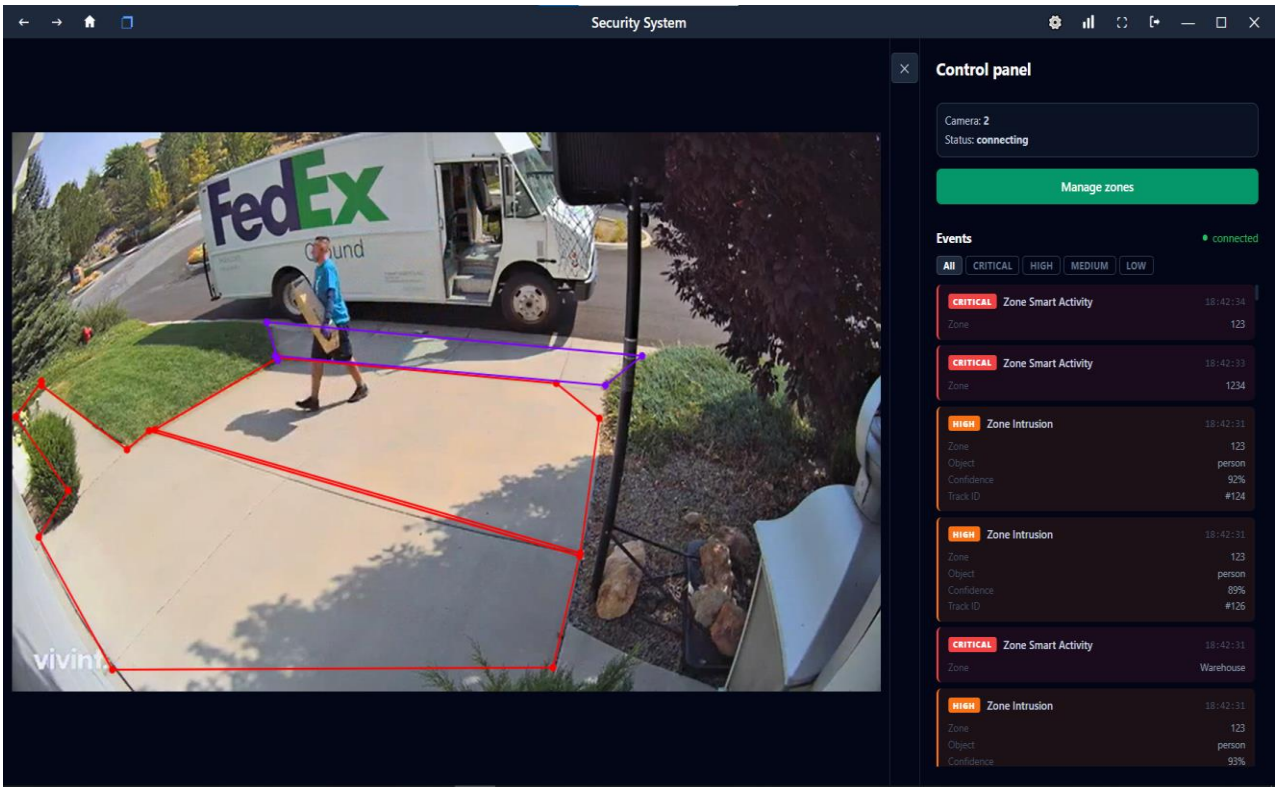


Рисунок 3.3 – Екран моніторингу камери з панеллю подій

Сторінка детального аналізу, наведена на рисунку 3.3, відображає вибраний відеопотік на всю доступну площу інтерфейсу із збереженням пропорцій кадру. Це запобігає спотворенню зображення і гарантує коректне відображення результатів відеоаналітики поверх відео.

На зображенні відображаються зони спостереження у вигляді векторних полігонів. Для зручності використовується кольорове розділення: червоний колір позначає зони обмеження, жовтий – периметр, зелений – безпечні ділянки, а фіолетовий використовується для транзитних зон і паркування. Якщо для певної зони налаштовано часове вікно неактивності, її контур відображається у вигляді напівпрозорого пунктиру, що показує зниження рівня контролю.

Праворуч розміщена бічна панель, яка містить інформацію про камеру, стан з’єднання та журнал подій. Події відображаються у вигляді списку з основними параметрами, такими як тип об’єкта, рівень впевненості та інші характеристики, з можливістю швидкого перегляду та фільтрації. Окрім цього, панель містить окремий режим перегляду зон, у якому користувач може переглядати налаштовані зони та змінювати їх параметри. Доступ до редагування зон надається лише користувачам з правами адміністратора, оператор має можливість лише перегляду.

Розділ конфігурації системи, зображений на рисунку 3.4, побудовано на основі вертикальної системи вкладок. Видимість окремих розділів визначається роллю користувача: повний доступ до налаштувань має адміністратор, тоді як оператор може переглядати відомості про додані камери і змінювати лише параметри, що стосуються відображення інтерфейсу.

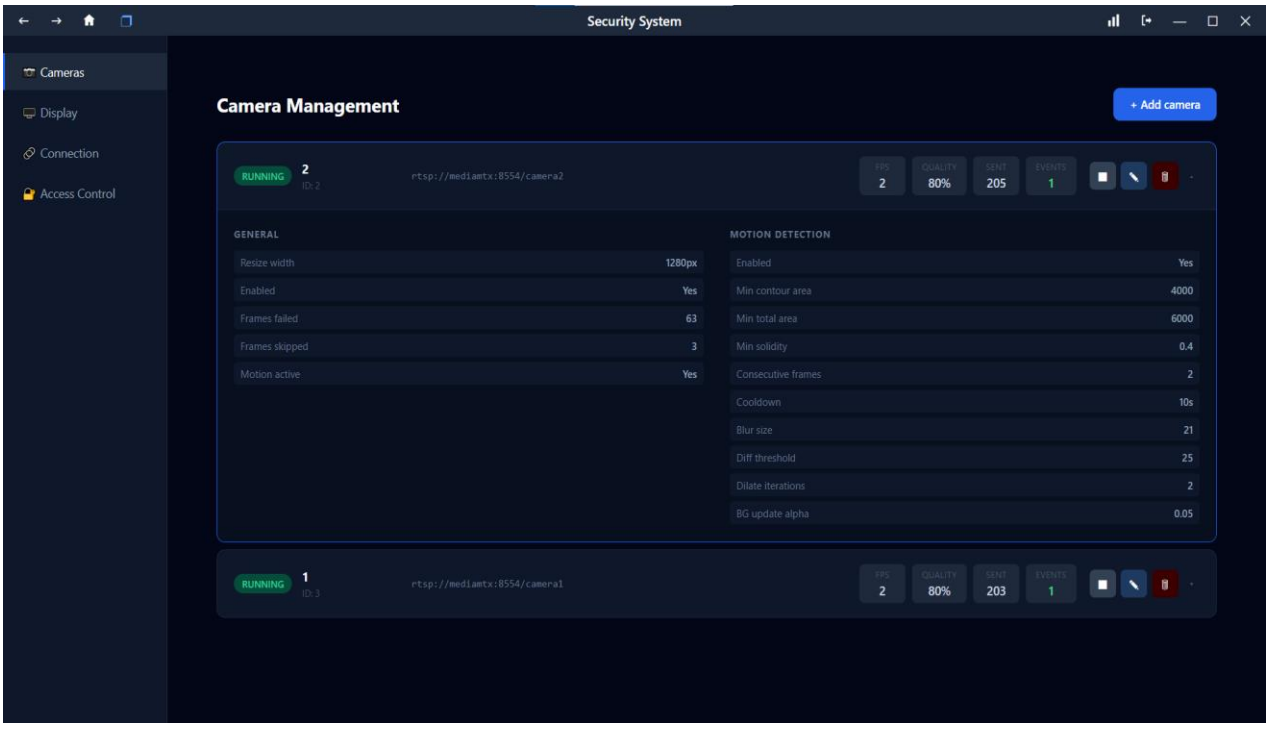


Рисунок 3.4 – Екран налаштувань, вкладка керування камерами

У вкладці керування камерами (рис. 3.4) відображається перелік підключених пристроїв. У згорнутому стані картка пристрою містить базову інформацію про камеру, зокрема мережеву адресу, частоту кадрів, рівень

компресії та кількість зафіксованих подій. Розгортання картки відкриває доступ до детальних параметрів та налаштувань відеоаналітики. Додавання нової камери реалізовано через модальне вікно, яке містить як базові параметри підключення, так і додаткові налаштування.

Вкладка відображення призначена для налаштування робочого простору оператора. Тут можна змінювати структуру сітки камер, встановивши від 4 до 16 слотів та призначати конкретні камери для кожної позиції. Налаштування зберігаються окремо для кожного вікна, що забезпечує використання кількох незалежних конфігурацій на різних моніторах.

Вкладка підключення, доступна лише адміністратору, використовується для налаштування адрес серверних сервісів. Зокрема, тут задається URL основного API та адреса сервера потокового відео, що використовується для відтворення відеопотоків.

У вкладці керування доступом реалізовано базові можливості адміністрування користувачів. Вона містить форму створення нових облікових записів із вибором ролі, а також список існуючих користувачів із можливістю зміни їх паролів.

Дашборд аналітики, поданий на рисунках 3.5 та 3.6, використовується для аналізу подій, що були зафіксовані системою. Він об’єднує кілька візуальних елементів, які дозволяють швидко оцінити загальний стан безпеки. Для зручності користувача передбачено селектор відеоджерела, який забезпечує безшовне перемикатися між загальною статистикою системи та даними по окремій камері.

У верхньому блоці зібрано п’ять інформаційних карток із ключовими метриками: сумарна кількість подій, кількість критичних та високих загроз, статистика активних зон та загальний стан відеомережі. У центральній області розміщено кілька графічних візуалізацій. Індикатор рівня ризику виводить динамічне значення, яке обчислюється на базі останніх подій. Стовпчикова та кругова діаграми показують розподіл загроз за рівнем критичності та типом. Рейтинг тривожних зон представлено у вигляді горизонтальних індикаторів, що відображають кількість спрацьовувань у кожній зоні.

Рисунок 3.5 – Екран аналітики, верхня частина

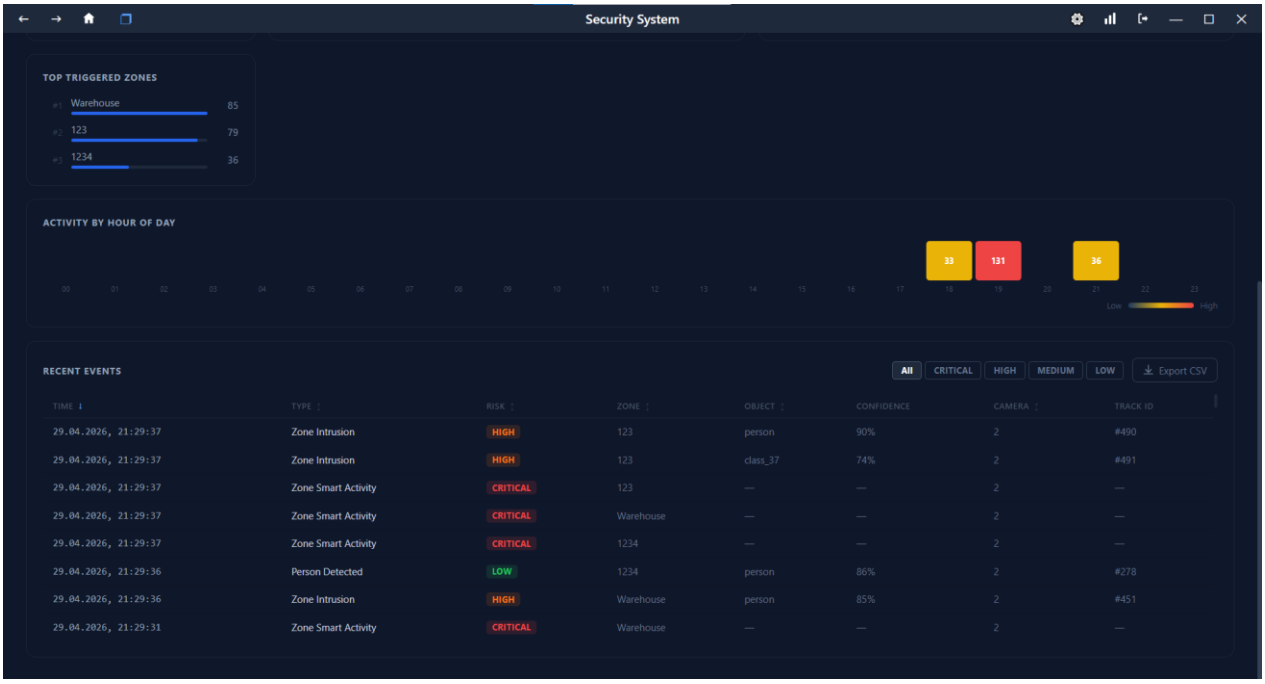


Рисунок 3.6 – Екран аналітики, нижня частина

Окремо реалізовано теплову карту активності, яка візуалізує пікові періоди навантаження на систему безпеки. У нижній частині сторінки розміщено інтерактивну таблицю подій із можливістю сортування за кількома параметрами та експорту даних.

Комбінація візуальних графіків, теплової карти та деталізованого журналу подій надає хороший інструментарій для розслідування інцидентів. Візуальне

розділення загальної статистики та конкретних логів у межах одного екрана виключає необхідність перемикання між різними вкладками чи зовнішніми програмами. Це зберігає концентрацію уваги оператора під час роботи з архівом.

Детальний вигляд усіх панелей конфігурації, включаючи модальні вікна додавання камер, керування користувачами та розширені налаштування зон безпеки, представлено у додатку Б (рисунки Б.1-Б.7).

3.3 Реалізація відображення відеоаналітики та подій безпеки

Ефективність системи відеомоніторингу визначається не лише точністю алгоритмів виявлення об’єктів, а й тим, наскільки зрозуміло та зручно результати цієї обробки відображаються у клієнтському інтерфейсі. Тому реалізація цієї підсистеми є одним із ключових завдань розробки клієнтської частини.

Ця підсистема об’єднує відтворення медіапотоку в режимі реального часу, накладання результатів аналітики безпосередньо на кадр та структуроване виведення інцидентів безпеки. Логіка роботи з відео та графічними елементами реалізована в окремому компоненті `CameraTile`, тоді як управління зонами та обробка потоку подій виконується на рівні батьківського компонента сторінки моніторингу. Програмну реалізацію компонента з логікою динамічного рендерингу зон подано в Додатку Б (лістингу В.2).

Відеопотік від камери транслюється через сервер MediaMTX за протоколом WebRTC та відтворюється у клієнтському застосунку за допомогою вбудованого HTML-елемента `iframe`, який підключається до відповідного потокового ресурсу. Адреса потоку формується динамічно на основі RTSP-посилання вибраної камери, що дозволяє гнучко підключати різні відеоджерела. Відео займає всю доступну область компонента із збереженням початкового співвідношення сторін, що забезпечує коректне відображення без спотворень під час зміни розмірів вікна застосунку.

Поверх відеотрансляції з абсолютним позиціюванням розміщується графічне полотно `canvas`, яке використовується для відображення аналітичних

зон. Важливо, щоб розміри цього полотна повністю співпадали з розмірами відеоелемента, оскільки від цього залежить коректність накладання графіки на зображення. Для забезпечення цього використовується інтерфейс `ResizeObserver`. Він автоматично відстежує будь-які зміни розмірів відеоблоку та синхронізує розміри графічного шару (ліст. 3.3).

Лістинг 3.3 – Синхронізація розмірів canvas з відеоелементом

```
useLayoutEffect(() => {
        const media = mediaRef.current;
        const canvas = canvasRef.current;
        if (!media || !canvas) return;
        const updateSize = () => {
            const { width, height } = media.getBoundingClientRect();
            canvas.width = width;
            canvas.height = height;
            setCanvasSize({ width, height });
        };
        const resizeObserver = new ResizeObserver(() => updateSize());
        resizeObserver.observe(media);
        updateSize();
        return () => resizeObserver.disconnect();
    }, [isFocused, isPanelOpen]);
```

кінець лістингу 3.3

Координати вершин тривожних зон зберігаються у нормалізованому вигляді. Кожне значення лежить у діапазоні від нуля до одиниці та фіксує відносне положення точки щодо ширини або висоти кадру. Цей архітектурний підхід гарантує абсолютну незалежність геометрії зон від роздільної здатності відеопотоку чи розмірів монітора. Під час рендерингу нормалізовані координати конвертуються в абсолютні піксельні значення шляхом множення на поточні розміри полотна. Алгоритм цієї трансформації наведено у лістингу 3.4.

Для зручності сприйняття кожному типу зони призначено окремий колір: червоний виділяє території обмеженого доступу, жовтий маркує периметр, зелений позначає безпечні ділянки, фіолетовий закріплено за зонами входу та паркування, а блакитний ідентифікує пішохідні маршрути.

Для зручності сприйняття кожному типу зони призначено окремий колір: червоний виділяє території обмеженого доступу, жовтий маркує периметр, зелений позначає безпечні ділянки, фіолетовий закріплено за зонами входу та паркування, а блакитний ідентифікує пішохідні маршрути.

Лістинг 3.4 – Перетворення нормалізованих координат зони у пікселі

```
ctx.moveTo(zone.points[0][0] * width, zone.points[0][1] * height);

for (let i = 1; i < zone.points.length; i++) {
     ctx.lineTo(zone.points[i][0] * width, zone.points[i][1] * height);
}
```

кінець лістингу 3.4

У системі також реалізовано підтримку часових вікон активності – заздалегідь визначених інтервалів, протягом яких зона тимчасово переходить у розслаблений режим зі зниженим рівнем контролю. Перевірка активності часового вікна виконується локально на стороні клієнта шляхом порівняння поточного системного часу з заданими інтервалами. Цикл перевірки відбувається щосекунди. Якщо зона переходить у неактивний стан, її контур відображається напівпрозорою пунктирною лінією, що дозволяє оператору одразу побачити зміну режиму. Алгоритм перевірки активності часового вікна наведено у лістингу 3.5.

Лістинг 3.5 – Перевірка активності часового вікна зони

```
function isZoneRelaxed(zone, now) {
   if (!zone.time_windows || zone.time_windows.length === 0) return
 false;
   const currentMinutes = now.getHours() * 60 + now.getMinutes();
   return zone.time_windows.some(tw => {
        if (!tw.start || !tw.end) return false;
        const [sh, sm] = tw.start.split(":").map(Number);
        const [eh, em] = tw.end.split(":").map(Number);
        return currentMinutes >= sh * 60 + sm &&
               currentMinutes <= eh * 60 + em;
        });
}
```

кінець лістингу 3.5

Керування потоком інцидентів реалізовано за допомогою спеціалізованого хука `useEventStream`, повний код якого наведено в додатку В (лістинг В.3). Під час ініціалізації компонент завантажує історію подій через REST API, гарантуючи оператору доступ до історії одразу після завантаження сторінки. Паралельно встановлюється двостороннє WebSocket-з'єднання для безперервного отримання нових подій у режимі реального часу. У разі втрати з'єднання спрацьовує вбудований механізм автоматичного відновлення з'єднання із затримкою у кілька секунд.

Усі вхідні повідомлення піддають попередній обробці: алгоритм перевіряє наявність обов'язкових полів та зводить їх до єдиного формату. Цей етап є критично необхідним через можливі структурні розбіжності між відповідями REST та пакетами WebSocket. Додатково реалізовано перевірку на дублікати, яка запобігає повторному додаванню однакових подій. Після цього інциденти фільтруються за ідентифікатором камери та відображаються у журналі.

Візуальне подання інциденту формується у вигляді картки, яка містить кольоровий індикатор рівня ризику, тип загрози, часову мітку та деталізовану інформацію: назву зони, клас виявленого об'єкта, точність моделі розпізнавання та унікальний ідентифікатор треку. Для зручності сприйняття використовується кольорове кодування: червоний колір відповідає критичним загрозам, помаранчевий – високому рівню ризику, жовтий – середньому, а зелений – низькому. Інтегрована система фільтрації допомагає оператору зосередитися на найбільш важливих інцидентах. Актуальний стан мережевого підключення транслюється через відповідний статус-індикатор у верхній частині панелі.

Отже, реалізована підсистема відображення відеоаналітики формує надійне синхронізоване середовище, де трансляція медіаданих безшовно поєднується з векторною графікою та структурованим потоком інцидентів. Це створює потужний інформаційний фундамент для блискавичного реагування персоналу на будь-які загрози безпеці.

3.4 Реалізація дашбордів і журналу подій

Для аналізу стану безпеки об’єкта в клієнтській частині системи реалізовано окрему сторінку аналітики, яка поєднує інтерактивні дашборди та журнал інцидентів. На відміну від сторінки моніторингу, що орієнтована на спостереження в реальному часі, ця сторінка призначена для узагальнення накопичених даних і виявлення закономірностей у роботі системи.

Аналітичні компоненти сторінки використовують єдине джерело даних – масив подій, що формується хуком `useEventStream`. Завдяки цьому не виникає потреби у додаткових запитах до серверного API, оскільки аналітика будується на основі подій, отриманих під час фонової роботи застосунку. При надходженні нових повідомлень через WebSocket графіки оновлюються автоматично. Кількість подій, які зберігаються у пам’яті, обмежена 200 записами. У разі перевищення цього ліміту найстаріші події поступово замінюються новими.

Для зручності аналізу впроваджено фільтрацію за камерою через селектор у верхній частині сторінки (рис. 3.5). При переході зі сторінки моніторингу конкретної камери її ідентифікатор автоматично передається через параметр URL і підставляється у фільтр. Це забезпечує логічний перехід між сторінками без необхідності повторного вибору камери.

У верхній частині дашборду реалізовано блок із п’яти інформаційних карток, що відображають ключові метрики системи. Значення цих показників формуються на основі масиву подій і включають загальну кількість інцидентів у поточній сесії, кількість критичних і високих загроз, кількість унікальних зон спрацьовування, а також співвідношення активних камер до їх загальної кількості. Оновлення даних відбувається автоматично при зміні масиву інцидентів, що забезпечує актуальність відображуваної інформації.

Центральним елементом аналітичного простору є динамічний індикатор агрегованого рівня ризику, який відображає загальний стан безпеки об’єкта у вигляді числового значення від 0 до 100. Показник обчислюється на основі

останніх двадцяти подій із застосуванням вагових коефіцієнтів відповідно до рівня небезпеки кожного інциденту за виразом 3.1.

$$R = \min\left(100,\ \frac{\sum_{i=1}^{N} w_i}{N}\right), \tag{3.1}$$

де R – агрегований показник ризику, N – кількість останніх подій (не більше 20), w_i – вага i-ї події: 100 для critical, 60 для high, 30 для medium, 10 для low.

Реалізацію цього обчислення на стороні клієнтського застосунку наведено у лістингу 3.6.

Лістинг 3.6 – Обчислення агрегованого показника ризику

```
const weights = { critical: 100, high: 60, medium: 30, low: 10 };
const recent = events.slice(0, 20);
const total = recent.reduce((sum, e) => sum + (weights[e.risk_level] ||
0), 0);
return Math.min(100, Math.round(total / recent.length));
```

кінець лістингу 3.6

Отримане значення R заокруглюється до цілого значення та інтерпретується за чотирма градаціями: $R < 25$ – низький рівень небезпеки, $25 \le R < 50$ – середній, $50 \le R < 75$ – високий, $R \ge 75$ – критичний. Колір індикатора автоматично адаптується відповідно до поточної градації, сприяючи миттєвому візуальному сприйняттю загального стану безпеки.

Розподіл подій за рівнями ризику відображається у вигляді стовпчикової діаграми з відповідним кольоровим кодуванням. Для аналізу інцидентів за їх типами використовується кільцева діаграма, яка показує співвідношення між різними категоріями подій, зокрема вторгненнями у зони, виявленням осіб та фіксацією руху. Побудова цих візуалізацій здійснюється за допомогою бібліотеки Recharts, що забезпечує адаптивне масштабування елементів та відображення інтерактивних підказок при наведенні курсора [21].

Рейтинг найбільш активних тривожних зон подається у вигляді горизонтальних смугових індикаторів з нормованими значеннями. Кожен рядок містить назву зони та кількість спрацьовувань, а довжина смуги відображає відносну активність ділянки порівняно з лідером рейтингу.

Для аналізу часових закономірностей у роботі системи реалізовано теплову карту активності по годинах доби. Графік складається з 24 клітинок, кожна з яких відповідає окремій годині. Інтенсивність кольору клітинки прямо пропорційна кількості подій, зафіксованих у відповідний часовий проміжок: від темно-синього при низькій активності до яскраво-червоного при максимальній. Такий інструмент допомагає адміністратору системи визначити пікові години навантаження та оптимізувати графік чергування операторів відповідно до реальної статистики.

У нижній частині сторінки аналітики (рис. 3.6) розміщено інтерактивний журнал подій. Таблиця виводить до 200 найновіших записів та підтримує багатофакторне сортування за часом, типом події, рівнем ризику, зоною, класом об’єкта або джерелом. Оновлення таблиці відбувається в режимі реального часу при надходженні нових подій. Фільтрація за рівнем критичності виконується через панель швидких вкладок, розташованих над таблицею.

Для кожного інциденту відображаються: дата та час події у локальному часовому поясі, тип загрози, рівень ризику у вигляді кольорового маркера, назва зони, клас виявленої цілі, точність класифікації моделі, ідентифікатор камери та ідентифікатор треку об’єкта. У разі відсутності певного параметра система виводить прочерк.

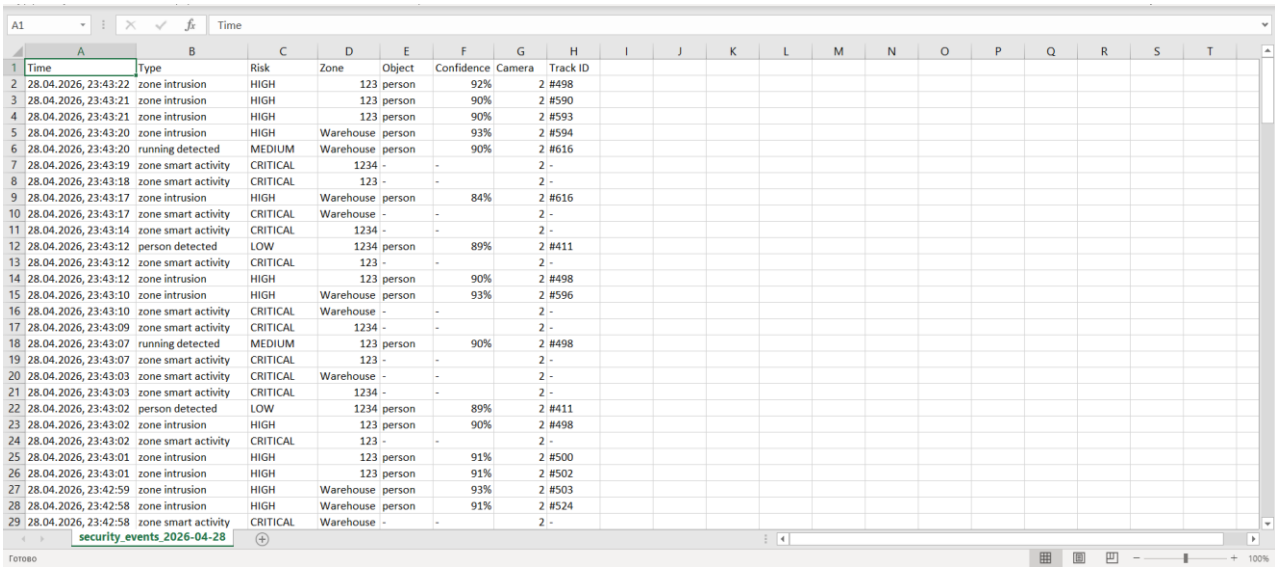
Окремою функціональною можливістю системи є функція експорту журналу подій у формат CSV для подальшого аналізу в зовнішніх редакторах таблиць. Формування файлу виконується на стороні клієнта без додаткових звернень до сервера. Для забезпечення коректного відкриття файлу в Microsoft Excel (рис. 3.7) використовується BOM-маркер і крапка з комою як роздільник полів, що дозволяє правильно відображати кирилицю та структуру даних.

Алгоритм формування та збереження документа продемонстровано у лістингу 3.7.

Лістинг 3.7 – Формування та експорт журналу подій у формат CSV

```
const exportCSV = () => {
    const headers = ["Time", "Type", "Risk", "Zone", "Object",
 "Confidence", "Camera", "Track ID"];
    const csvContent = [
        headers.map(escape).join(";"),
        ...rows.map(r => r.join(";")),
    ].join("\r\n");
    const BOM = "\uFEFF";
    const blob = new Blob([BOM + csvContent], { type:
 "text/csv;charset=utf-8;" });
    const url = URL.createObjectURL(blob);
    const a = document.createElement("a");
    a.href = url;
        a.download = `security_events_${new
      Date().toISOString().slice(0, 10)}.csv`;
        a.click();
        URL.revokeObjectURL(url);
};
```

кінець лістингу 3.7



	Time	Type	Risk	Zone	Object	Confidence	Camera	Track ID
2	28.04.2026, 23:43:22	zone intrusion	HIGH	123	person	92%	2	#498
3	28.04.2026, 23:43:21	zone intrusion	HIGH	123	person	90%	2	#590
4	28.04.2026, 23:43:21	zone intrusion	HIGH	123	person	90%	2	#593
5	28.04.2026, 23:43:20	zone intrusion	HIGH	Warehouse	person	93%	2	#594
6	28.04.2026, 23:43:20	running detected	MEDIUM	Warehouse	person	90%	2	#616
7	28.04.2026, 23:43:19	zone smart activity	CRITICAL	1234	-	-	2	-
8	28.04.2026, 23:43:18	zone smart activity	CRITICAL	123	-	-	2	-
9	28.04.2026, 23:43:17	zone intrusion	HIGH	Warehouse	person	84%	2	#616
10	28.04.2026, 23:43:17	zone smart activity	CRITICAL	Warehouse	-	-	2	-
11	28.04.2026, 23:43:14	zone smart activity	CRITICAL	1234	-	-	2	-
12	28.04.2026, 23:43:12	person detected	LOW	1234	person	89%	2	#411
13	28.04.2026, 23:43:12	zone smart activity	CRITICAL	123	-	-	2	-
14	28.04.2026, 23:43:12	zone intrusion	HIGH	123	person	90%	2	#498
15	28.04.2026, 23:43:10	zone intrusion	HIGH	Warehouse	person	93%	2	#596
16	28.04.2026, 23:43:10	zone smart activity	CRITICAL	Warehouse	-	-	2	-
17	28.04.2026, 23:43:09	zone smart activity	CRITICAL	1234	-	-	2	-
18	28.04.2026, 23:43:07	running detected	MEDIUM	123	person	90%	2	#498
19	28.04.2026, 23:43:07	zone smart activity	CRITICAL	123	-	-	2	-
20	28.04.2026, 23:43:03	zone smart activity	CRITICAL	Warehouse	-	-	2	-
21	28.04.2026, 23:43:03	zone smart activity	CRITICAL	1234	-	-	2	-
22	28.04.2026, 23:43:02	person detected	LOW	1234	person	89%	2	#411
23	28.04.2026, 23:43:02	zone intrusion	HIGH	123	person	90%	2	#498
24	28.04.2026, 23:43:02	zone smart activity	CRITICAL	123	-	-	2	-
25	28.04.2026, 23:43:01	zone intrusion	HIGH	123	person	91%	2	#500
26	28.04.2026, 23:43:01	zone intrusion	HIGH	123	person	91%	2	#502
27	28.04.2026, 23:42:59	zone intrusion	HIGH	Warehouse	person	93%	2	#503
28	28.04.2026, 23:42:58	zone intrusion	HIGH	Warehouse	person	91%	2	#524
29	28.04.2026, 23:42:58	zone smart activity	CRITICAL	Warehouse	-	-	2	-

Рисунок 3.7 – Вигляд експортованих даних журналу подій у Microsoft Excel

Таким чином, розроблена підсистема дашбордів та журналу подій забезпечує комплексний інструментарій для аналізу стану безпеки. Вона

охоплює як обчислення агрегованих показників ризику, так і можливість експорту детальної статистики для подальшого опрацювання, що сприяє більш глибокому дослідженню інцидентів.

3.5 Тестування клієнтської частини системи

Тестування клієнтської частини системи відеомоніторингу проводилось методом ручного функціонального тестування з метою перевірки відповідності реалізованого функціонала вимогам, сформульованим у підрозділі 1.5. Перевірка здійснювалася за сценарним підходом із урахуванням як типових, так і граничних випадків використання. Тестуванню підлягали основні модулі застосунку: механізми автентифікації та розмежування доступу, відображення відеопотоків і зон спостереження, обробка та візуалізація подій безпеки, аналітичні дашборди, а також адміністративні функції.

На першому етапі перевірено коректність роботи механізму входу до системи при введенні коректних і некоректних облікових даних. У разі помилки інтерфейс відображає повідомлення з поясненням причини відмови. Після успішної автентифікації токен безпеки зберігається у локальному сховищі браузера, а користувач автоматично перенаправляється до захищеної частини застосунку. Додатково протестовано сценарій прямого переходу на закритий маршрут без попереднього входу – система блокує доступ і виконує перенаправлення на сторінку автентифікації, що виключає можливість несанкціонованого доступу.

Окрему увагу під час перевірки авторизації приділено тестуванню рольової моделі. Для користувача з роллю оператора обмежується доступ до вкладок підключення та керування користувачами, приховуються елементи редагування відеоджерел, а налаштування зон доступні лише у режимі перегляду. У режимі адміністратора відкривається повний набір функціональних можливостей системи. Також протестовано механізм міжвіконної синхронізації: виконання виходу з облікового запису в одному вікні призводить до завершення сеансу в

інших відкритих вікнах застосунку, що забезпечує узгодженість стану авторизації.

Наступним етапом стала перевірка модуля відображення відеопотоку та зон безпеки під час моніторингу цільової камери. Векторні полігони накладаються поверх відео камери з дотриманням заданої кольорової семантики. Протестовано логіку обробки часових вікон: при настанні заданого інтервалу контур зони автоматично змінюється у напівпрозору пунктирну лінію. Геометрична точність накладання результатів відеоаналітики перевірялася шляхом динамічної зміни розмірів вікна застосунку та взаємодії з бічною панеллю. Завдяки роботі алгоритму `ResizeObserver` графічний шар синхронізується з відеоелементом і зберігає правильне позиціонування аналітичних об’єктів незалежно від розмірів інтерфейсу.

Критично важливим етапом тестування виступала обробка подій безпеки та їх надходження до журналу бічної панелі. При надходженні нових інцидентів через WebSocket-з’єднання відповідні записи з’являються у списку без перезавантаження сторінки, що підтверджує роботу механізму оновлення в режимі реального часу. Перевірено функціональність фільтрації, яка забезпечує відображення подій за обраним рівнем ризику. На головному екрані підтверджено коректність візуальних індикаторів: камера, що зафіксувала інцидент, виділяється пульсуючим контуром та інформаційним маркером, що сприяє швидкій ідентифікації проблемного сектора оператором. При моделюванні втрати мережевого з’єднання система відображає відповідний статус, а механізм автоматичного відновлення повторно ініціалізує з’єднання із заданою затримкою без порушення роботи інтерфейсу.

Ефективність роботи аналітичного дашборду перевірено шляхом аналізу коректності обчислення основних метрик. Агрегований показник ризику, діаграми та рейтингові списки оновлюються реактивно при кожному новому інциденті. Додатково протестовано механізм передачі параметрів через URL: при переході до сторінки аналітики ідентифікатор камери автоматично застосовується для фільтрації даних. Функція експорту журналу подій формує

файл формату CSV, який коректно відкривається у табличних редакторах, зберігає структуру колонок і забезпечує правильне відображення кириличних символів.

Завершальною фазою стала перевірка адміністративних функцій. Перевірено повний цикл роботи з відеоджерелами: додавання нової камери, редагування параметрів та видалення пристрою. Також перевірено створення нових облікових записів користувачів і зміну їх даних. Окремо підтверджено стабільність роботи у багатовіконному режимі: кожне вікно застосунку зберігає власну конфігурацію відображення камер, що унеможливлює конфлікти між різними робочими середовищами.

За результатами проведеного функціонального тестування встановлено, що реалізована клієнтська частина системи відповідає основним функціональним вимогам та забезпечує стабільну роботу інтерфейсу. Система коректно обробляє дії користувача, підтримує відображення відеопотоків і результатів відеоаналітики, забезпечує обробку та візуалізацію інцидентів у режимі реального часу, а також реалізує розмежування доступу відповідно до ролей. Окремо підтверджено працездатність аналітичного модуля, механізмів експорту даних і адміністративних інструментів керування, що свідчить про цілісність та узгодженість роботи всіх компонентів клієнтського застосунку.

ЗАГАЛЬНІ ВИСНОВКИ І РЕКОМЕНДАЦІЇ

У результаті виконання кваліфікаційної роботи було розроблено клієнтську частину інформаційної системи відеомоніторингу, призначену для відображення відеопотоків, обробки подій безпеки та забезпечення взаємодії оператора із системою в режимі реального часу.

Під час виконання роботи було проаналізовано сучасні системи моніторингу безпеки, досліджено особливості веб-орієнтованих рішень та визначено основні вимоги до користувацького інтерфейсу систем відеомоніторингу. За результатами проведеного дослідження сформульовано наступні висновки:

1) проведено обґрунтування вибору технологічного стеку для реалізації клієнтської частини системи. Для побудови інтерфейсу обрано бібліотеку React, для централізованого управління станом застосунку – Redux Toolkit, а для створення настільного середовища виконання – Electron. Використання такого поєднання технологій дозволяє забезпечити високу швидкість оновлення інтерфейсу, модульність та можливість подальшого розширення функціоналу системи;

2) спроектовано архітектуру клієнтської частини застосунку з урахуванням вимог до роботи в режимі реального часу. Запропонована архітектура базується на компонентному підході та багатопроцесній моделі Electron, що дозволяє відокремити інтерфейс користувача від системних процесів і підвищити стабільність роботи програмного забезпечення;

3) розроблено структуру користувацького інтерфейсу для ролей оператора та адміністратора. При проектуванні інтерфейсу враховано принципи мінімізації когнітивного навантаження, швидкого доступу до критично важливої інформації та підтримки ситуаційної обізнаності користувача під час роботи з системою;

4) обґрунтовано підхід до організації відображення відеопотоків у режимі реального часу. Для передачі відео запропоновано використання зв’язки RTSP та WebRTC із застосуванням сервера потоків MediaMTX, що дозволяє зменшити

затримки при передачі даних та забезпечити стабільне відтворення відео у клієнтському застосунку;

5) запропоновано підхід до реалізації аналітичних панелей та візуалізації подій безпеки. Використання графічних накладень, журналів подій та індикаторів рівня ризику дозволяє підвищити наочність представлення інформації та спростити процес аналізу інцидентів оператором системи;

6) обґрунтовано методи інтеграції клієнтської частини із серверною інфраструктурою. Для передачі конфігураційних даних використано REST API, а для отримання подій у режимі реального часу – WebSocket-з'єднання. Такий підхід забезпечує ефективний розподіл навантаження та підтримує стабільну взаємодію між компонентами системи;

7) проведений аналіз запропонованих архітектурних та технологічних рішень показав, що обраний підхід відповідає вимогам до сучасних систем відеомоніторингу. Використані технології дозволяють забезпечити масштабованість, гнучкість та можливість подальшого розвитку системи залежно від потреб конкретного об'єкта.

Для подальшого розвитку та вдосконалення системи доцільно рекомендувати:

– розширення функціоналу відеоаналітики за рахунок інтеграції додаткових моделей комп'ютерного зору;

– реалізацію механізмів резервування каналів передачі даних та автоматичного відновлення з'єднання у разі мережевих збоїв;

– оптимізацію обробки великої кількості одночасних відеопотоків;

– впровадження системи централізованого журналювання та моніторингу помилок;

– впровадити централізоване сховище для довготривалого зберігання відеозаписів та історії подій безпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Barzantny C., Bruder C. Measuring Situation Awareness in Control Room Teams. Engineering Psychology and Cognitive Ergonomics. Cognition and Design. Cham, 2020. URL: https://link.springer.com/chapter/10.1007/978-3-030-49183-3_1 (access date: 10.02.2026).

2. Cabanillas-Carbonell M. et al. Artificial intelligence in video surveillance systems for suspicious activity detection and incident response. Revista Facultad de Ingeniería. 2024. URL: https://surl.li/vfhdjf (access date: 10.02.2026).

3. Micron Technology. Securing Your Business While Lowering TCO with VSaaS. ASMAG Global Security Research. 2024. URL: https://www.asmag.com/download/micron_lowering_tco_with_vsaas.pdf (access date: 17.02.2026).

4. Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446. 2018. URL: https://datatracker.ietf.org/doc/html/rfc8446 (access date: 05.03.2026).

5. Choose between traditional web apps and Single Page Apps (SPAs). Microsoft Learn: Architecture Guide. 2024. URL: https://surl.li/xugfpj (access date: 05.02.2026).

6. Wang C.-Y., Bochkovskiy A., Liao H.-Y. M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv preprint arXiv:2207.02696. 2022. URL: https://arxiv.org/abs/2207.02696 (access date: 10.02.2026).

7. Oladimeji O. O. et al. Data-Driven Project Monitoring: Leveraging Dashboards and KPIs to Track Performance in Technology Implementation Projects. Frontiers in Multidisciplinary Research. 2025. Vol. 2. P. 30-45. URL: https://surl.lu/ogqapv (access date: 16.02.2026).

8. ISO 9241-210:2019. Ergonomics of human-system interaction – Part 210: Human-centered design for interactive systems. Geneva: International Organization for

Standardization, 2019. URL: https://www.iso.org/standard/77520.html. (access date: 10.02.2026).

9. Milestone Systems | Global leading video management software provider. Milestone Systems. URL: https://www.milestonesys.com/ (access date: 16.02.2026).

10. Security Systems for the Modern Enterprise | Verkada. URL: https://www.verkada.com/ (access date: 16.02.2026).

11. Ajax Security – Alarms & Smart Systems | Official website. Ajax Systems. URL: https://ajax.systems/ (access date: 16.02.2026).

12. React. React. URL: https://react.dev/ (access date: 03.02.2026).

13. Redux Toolkit. Redux Toolkit. URL: https://redux-toolkit.js.org/ (access date: 27.02.2026).

14. Build cross-platform desktop apps with JavaScript, HTML, and CSS | Electron. URL: https://www.electronjs.org/ (access date: 10.02.2026).

15. MediaMTX | ready-to-use, open source, live media router. MediaMTX | ready-to-use, open source, live media router. URL: https://mediamtx.org/ (access date: 07.03.2026).

16. WebSocket API (WebSockets) - Web APIs | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API (access date: 05.03.2026).

17. Vite. vitejs. URL: https://vite.dev/ (access date: 03.02.2026).

18. Sass: Syntactically Awesome Style Sheets. Sass: Syntactically Awesome Style Sheets. URL: https://sass-lang.com/ (access date: 03.02.2026).

19. RFC 2326: Real Time Streaming Protocol (RTSP). IETF Datatracker. URL: https://datatracker.ietf.org/doc/html/rfc2326 (access date: 07.03.2026).

20. WebRTC. URL: https://webrtc.org/ (access date: 07.03.2026).

21. Recharts: A composable charting library built on React components. URL: https://recharts.org/ (access date: 10.04.2026).

22. Терлецький Т. В. , Кайдик О. Л. Кваліфікаційна робота: методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Інформаційні системи

та технології охорони і безпеки» галузі знань 12 Інформаційні технології спеціальності 126 Інформаційні системи та технології денної та заочної форм навчання. Луцьк: ЛНТУ, 2025. 53 с.