

Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”

Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор

_____ Олексій СМІРНОВ

“_____” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему

**“Програмне забезпечення системи відеоспостереження на основі технології
FRIGATE NRV”**

Виконав здобувач вищої освіти

IV курсу, групи КІ-22-1

ОПП «Комп’ютерна інженерія»

спеціальності 123 «Комп’ютерна інженерія»

_____ Р.П. Дяченко

«_____» _____ 2026 р.

Керівник проекту

кандидат технічних наук, доцент

_____ Р. М. Минайленко

«_____» _____ 2026 р.

Рецензент _____

Центральноукраїнський національний технічний університет
Факультет Механіко-технологічний

Кафедра Кібербезпеки та програмного забезпечення

Рівень вищої освіти бакалавр

Галузь знань ._12 _____ “Інформаційні технології” _____

Спеціальність 123 “Комп’ютерна інженерія”

Освітньо-професійна (освітньо-наукова) програма _ “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ

Завідувач кафедри

д.т.н., проф.

_____ Олексій СМІРНОВ

“ _____ ” _____ 20__ року

**ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА
ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ
ЗДОБУВАЧА ВИЩОЇ ОСВІТИ**

Дяченку Руслану Павлович

(прізвище, ім’я, по батькові)

1. Тема роботи Програмне забезпечення системи відеопостереження на основі технології FRIGATE NRV

2. Керівник роботи Минайленко Роман Миколайович, канд. техн. наук, доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “27” 02 2026 року №133-02

3. Строк подання роботи до захисту _____ р.

4. Мета та завдання випускної кваліфікаційної роботи Метою роботи є розробка програмного забезпечення системи відеоспостереження на основі технології Frigate NRV

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) _____

_____ 1. Призначення та область використання. _____

_____ 2.Перегляд аналогічних існуючих систем. _____

_____ 3. Опис і обґрунтування проектних рішень. _____

_____ 4. Етапи програмування системи. _____

_____ 5. Впровадження системи в промислову експлуатацію _____

_____ 6. Висновки. _____

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи _____ 1 аркуш _____

Функціональна схема системи

1 аркуш

Діаграма процесів

1 аркуш

Блок-схема алгоритму роботи додатку

2 аркуша

7. Дата видачі завдання “27”_02_2026 року

К6ПЗ_2026

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем керування	10.03.2026 р.	
2.	Постановка задачі, оформлення ТЗ	16.03.2026 р.	
3.	Розробка моделі компонент	21.03.2026 р.	
4.	Розробка структур даних	30.03.2026 р.	
5.	Розробка алгоритмів зв'язку та відображення	08.03.2026 р.	
6.	Програмування алгоритмів	18.04.2026 р.	
7.	Оформлення ПЗ	27.04.2026 р.	
8.	Попередній захист роботи	16.05.2026 р.	

Дата видачі завдання
«27» 02 2026 року

Підпис керівника

(прізвище та ініціали)

Завдання прийнято до виконання
« » 2026 р.

Підпис здобувача

АНОТАЦІЯ

Дяченко Р. П. Програмне забезпечення системи відеоспостереження на основі технології Frigate NVR. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В даній кваліфікаційній бакалаврській роботі розроблено програмне забезпечення для інтелектуального відеомоніторингу, що використовує можливості периферійних обчислень та штучного інтелекту.

Система базується на технології Frigate NVR та забезпечує аналіз відеопотоків у реальному часі з мінімальним навантаженням на центральний процесор завдяки використанню апаратних прискорювачів.

Метою розробки є створення аналітичної платформи, здатної самостійно ідентифікувати об'єкти, здійснювати їх дедуктивний аналіз та синтезувати отримані дані у текстові звіти за допомогою локальних мовних моделей (LLM).

Результат роботи — програмна реалізація системи, що включає асинхронний бекенд на FastAPI, базу даних часових рядів TimescaleDB та інтелектуальний модуль ідентифікації об'єктів на основі векторних ембедінгів у ChromaDB.

В процесі роботи проведено порівняльний аналіз існуючих аналогів (Blue Iris, Scrypted та ін.), обґрунтовано вибір архітектури YOLO для детекції та протоколу MQTT для передачі телеметрії.

Повною мірою описані компоненти системи, архітектура баз даних та алгоритми взаємодії процесів. Розроблено зручний веб-інтерфейс на React.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11 або Linux у середовищі Docker.

Ключові слова: комп'ютерна інженерія, Frigate NVR, штучний інтелект, комп'ютерний зір, YOLO, периферійні обчислення, FastAPI, дедуктивний аналіз.

ABSTRACT

Diachenko R. P. Software for a Video Surveillance System Based on Frigate NVR Technology. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

This bachelor's thesis describes the development of software for intelligent video monitoring that leverages edge computing and artificial intelligence capabilities. The system is based on Frigate NVR technology and provides real-time video stream analysis with minimal CPU load by utilizing hardware accelerators.

The aim of the development is to create an analytical platform capable of autonomous object identification, deductive analysis, and synthesis of the acquired data into text reports using local large language models (LLMs).

The result of the work is a software implementation of a system that includes an asynchronous backend developed with FastAPI, a TimescaleDB time-series database, and an intelligent object identification module based on vector embeddings in ChromaDB.

In the course of the work, a comparative analysis of existing analogues (Blue Iris, Scrypted, etc.) was conducted, and the choice of the YOLO architecture for detection and the MQTT protocol for telemetry transmission was justified. The components of the system, the database architecture, and the process interaction algorithms are fully described. A user-friendly web interface was developed using React. The program can be used on PCs running Windows 10/11 or Linux within a Docker environment.

Keywords: computer engineering, Frigate NVR, artificial intelligence,

c
o
m
p
u
t
e

К6ПЗ-2026

Вим.	Арк.	№ документ	Підп.	Дата	ВКРБ-123.26.0010.00.00.ПЗ				
Розробив	Дяченко Р.П.				Програмне забезпечення системи відеоспостереження на основі технології FRIGATE NRV	Літ.	Аркуш	Аркушів	
Перевірів	Минайленко Р.М							57	
Н. Конт. р.	Коваленко А.С.					ЦНТУ KI-22-1			
Зам.	Сейменов О.А.								

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ	3
ВСТУП.....	4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ.....	5
1.1 Призначення системи.....	5
1.2 Область застосування.....	7
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ.....	9
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.....	9
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування	15
2.3 Розгорнута постановка завдання.....	19
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ.....	21
3.1 Опис функціонування системи	21
3.2 Розробка структурної схеми.....	27
3.3 Розробка функціональної схеми.....	32
3.4 Розробка діаграми процесів.....	36
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРЕМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ.....	39
4.1 Розробка блок-схем та опис алгоритмів функціонування системи.....	39
4.2 Захист розробленого програмного забезпечення	48
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ	50
6 ОСНОВНІ ВИСНОВКИ.....	53
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	54

Вим.	Арк.	№ документ	Підп.	Дата
Розробив		Дяченко Р.П.		
Перевірів		Минайленко Р.М		
Н. Конт. р.		Коваленко А.С.		
Зам.		Сейменов О.А.		

ВКРБ-123.26.0010.00.00.ПЗ

Програмне забезпечення системи
відеоспостереження на основі
технології FRIGATE NRV

Літ.	Аркуш	Аркушів
		57

ЦНТУ KI-22-1

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

CV	—	комп'ютерне розуміння візуальних образів.
AI	—	моделювання процесів людського інтелекту.
VMS	—	система управління відеопотоками.
NVR	—	пристрій мережевого відеозапису.
CNN	—	нейромережі для аналізу зображень.
LLM	—	мовна модель для генерації тексту.
API	—	інтерфейс взаємодії програмних компонентів.
TPU	—	тензорний прискорювач для нейронних мереж.
GPU	—	графічний процесор для складних обчислень.
YOLO	—	You Only Look Once, детекція об'єктів
Re-ID	—	повторна ідентифікація того самого об'єкта.
MQTT	—	протокол швидкої передачі повідомлень.
RTSP	—	поточкова трансляція відео в реальному часі.
Frigate NVR	—	інтелектуальна локальна відеоаналітика.
Embedding	—	векторний цифровий відбиток об'єкта.
Edge Computing	—	обробка даних на локальному пристрої.
Docker	—	платформа для запуску ізольованих контейнерів.

ВСТУП

Актуальність теми. Сучасні умови безпеки вимагають переходу від пасивного відеоспостереження до інтелектуальних аналітичних систем, що здатні діяти самостійно та в реальному часі. Використання технології Frigate NVR дозволяє вирішити проблему великої кількості хибних сповіщень та забезпечити високу швидкість детекції об'єктів за допомогою периферійних обчислень (Edge Computing). Розробка програмного забезпечення, що інтегрує векторний пошук та локальні мовні моделі для синтезу «дедуктивної історії» об'єктів, є актуальним завданням для автоматизації роботи служб безпеки та оптимізації аналізу великих масивів відеоданих.

Мета і завдання роботи. Метою роботи є розробка програмного забезпечення інтелектуальної системи відеоспостереження на основі технології Frigate NVR, яка забезпечує автоматизовану детекцію, ідентифікацію та дедуктивний аналіз подій у реальному часі.

Практичне значення одержаних результатів. Розроблене програмне забезпечення дозволяє суттєво скоротити час розслідування інцидентів завдяки автоматичній генерації текстових звітів про переміщення об'єктів. Система готова до впровадження на об'єктах цивільної та критичної інфраструктури, де вимагається висока конфіденційність даних, оскільки вся обробка інформації відбувається локально без передачі в хмарні сервіси. Матеріали роботи можуть бути використані як база для створення кастомних систем моніторингу в маркетингу, логістиці та системах «розумного будинку».

					ВКРБ-123.26.0010.00.00.ПЗ	Арк
						4
Вим	Арк.	№ докум.	Підп.	Дата		

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Сучасні умови безпеки та стрімкий розвиток інформаційних технологій вимагають радикального переходу від пасивного відеоспостереження до інтелектуальних аналітичних систем. Більшість традиційних систем відеоспостереження, які сьогодні масово використовуються на підприємствах та в приватних домогосподарствах, виконують виключно функцію відеореєстрації. Їхня головна проблема полягає в тому, що вони є реактивними - тобто використовуються постфактум, коли інцидент вже стався, і оператору доводиться витратити години на перегляд архіву в пошуках потрібного фрагмента.

Досліджуючи цю проблематику, я дійшов висновку, що навіть наявність базових детекторів руху в класичних камерах не вирішує проблему, а скоріше створює нову проблему хибні спрацювання. Тіні від дерев, зміна освітлення, дощ або комахи генерують сотні нерелевантних сповіщень, що призводить до так званої «втоми від тривоги» (alarm fatigue) у співробітників служб безпеки. Саме тому у своїй роботі я поставив за мету розробити програмне забезпечення, здатне діяти самостійно, аналізувати ситуацію в реальному часі та надавати користувачеві вже готову, синтезовану інформацію

Розроблене мною програмне забезпечення базується на технології Frigate NVR, інтелектуальній системі з відкритим кодом, яка використовує неймережі для аналізу кожного кадру відеопотоку. Використання цієї технології дозволило мені вирішити проблему великої кількості хибних сповіщень та забезпечити високу швидкість детекції об'єктів за рахунок периферійних обчислень (Edge Computing).

					ВКРБ-123.26.0010.00.00.ПЗ	Арк
						5
Вим	Арк.	№ докум.	Підп.	Дата		

Головним призначенням моєї системи є автоматизована детекція, ідентифікація та дедуктивний аналіз подій у реальному часі. Я спроектував програмне забезпечення не просто як засіб запису медіаданих, а як повноцінний аналітичний вузол, що перетворює "сирий" відеопотік на структуровану базу знань.

Призначення розробленого комплексу можна розділити на кілька ключових напрямків:

– **Інтелектуальна фільтрація та мінімізація хибних тривог:** Завдяки інтеграції моделей комп'ютерного зору (зокрема, архітектури YOLO), система записує відео та генерує сповіщення виключно при наявності в кадрі цільових, "живих" об'єктів (людей, автомобілів, тварин).

– **Оптимізація зберігання даних:** Я реалізував логіку, за якої постійний відеозапис у високій якості не ведеться, якщо в кадрі нічого не відбувається. Це дозволяє економити колосальні обсяги дискового простору та знижує навантаження на мережеву інфраструктуру.

– **Автоматизація розслідувань (Дедуктивна історія):** Найважливішим призначенням моєї системи є синтез «дедуктивної історії» об'єктів. За допомогою векторного пошуку (ChromaDB) та локальних великих мовних моделей (LLM Ollama) я створив інструмент, який самостійно агрегує розрізнені лог-файли з різних камер і формує зрозумілий текстовий звіт про переміщення об'єкта. Це дозволяє скоротити час розслідування інцидентів з годин до лічених хвилин.

– **Забезпечення повної конфіденційності:** Оскільки вся аналітика, включаючи роботу генеративного штучного інтелекту, виконується локально на Edge-обладнанні, система не потребує відправки чутливих даних (відео, фото облич, векторних ознак) у хмарні сервіси. Це є критично важливим для корпоративного сектору.

1.2 Область застосування

Розроблювана система може бути застосована в широкому спектрі галузей та організацій. Основні області застосування включають:

1. Об'єкти критичної інфраструктури та підприємства з підвищеними вимогами до безпеки. На заводах, електростанціях чи військових об'єктах використання хмарних технологій часто заборонено регламентами безпеки. Моя розробка ідеально підходить для таких умов, оскільки працює в ізольованому локальному середовищі (On-Premise). Здатність системи автоматично ідентифікувати персонал або транспортні засоби та миттєво сповіщати про появу невідомих об'єктів у закритих зонах (використовуючи гнучке налаштування «масок» та «зон») значно підвищує рівень фізичної кібербезпеки.

2. Роздрібна торгівля (Retail) та маркетинг. Хоча першочергова мета відеоспостереження - це охорона, я спроектував логіку збереження метаданих у базі TimescaleDB так, щоб її можна було використовувати для маркетингової аналітики. Область застосування тут включає:

- підрахунок унікальних відвідувачів за день;
- створення теплових карт переміщення клієнтів по торговому залу на основі зібраних координат;
- аналіз часу затримки покупців біля конкретних вітрин. За допомогою дедуктивного модуля система може надати звіт не просто про рух, а про те, скільки часу конкретний об'єкт (клієнт) взаємодіяв з певною зоною.

3. Логістика та складський облік. На великих складах відстеження переміщення вантажів або техніки є складною задачею. Інтеграція алгоритмів розпізнавання (Re-ID) на основі косинусної подібності векторів дозволяє моїй системі "вести" об'єкт від камери до камери. Якщо навантажувач або палета зникає з поля зору однієї камери і з'являється на іншій, система здатна об'єднати ці події в єдиний хронологічний ланцюжок, полегшуючи диспетчеризацію.

					ВКРБ-123.26.0010.00.00.ПЗ	Арк
						7
Вим	Арк.	№ докум.	Підп.	Дата		

4. **Екосистеми «Розумного будинку» (Smart Home).** Завдяки нативній підтримці протоколу MQTT, розроблена мною система легко інтегрується з такими платформами, як Home Assistant або Node-RED. У цій області використання програмне забезпечення виступає як найсучасніший інтелектуальний сенсор. Наприклад, замість того, щоб просто вмикати світло на рух гілки, система може увімкнути освітлення на подвір'ї та надіслати сповіщення на смартфон лише тоді, коли детектовано людину або автомобіль, ігноруючи при цьому домашніх тварин.

Отже, розроблене мною програмне забезпечення виходить далеко за рамки традиційного NVR (Network Video Recorder). Його призначення - це комплексний інтелектуальний аналіз ситуації, формування векторних баз ознак об'єктів та надання користувачеві зручного інструменту взаємодії з архівами відеоданих через природну мову, за допомогою ШІ.

Область використання охоплює практично всі сфери, де потрібен комп'ютерний зір, миттєва реакція системи «тут і зараз» на різні події, а також глибока ретроспективна аналітика без порушення правил конфіденційності. Впровадження цієї системи здатне радикально змінити підхід до моніторингу, звільнивши людський ресурс від рутинного перегляду відео на користь прийняття управлінських та безпекових рішень на основі готових аналітичних звітів.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Для об'єктивного оцінювання доцільно розглянути аналоги Frigate NVR: Blue Iris, Scrypted, NX Witness та Synology Surveillance Station (див. Рисунок 2.1). Після їх огляду буде сформовано порівняльну таблицю за критеріями, що відповідають вимогам практичного застосування в бізнесі.



Рисунок 2.1 — Логотип Frigate NVR та його аналогів

Перед тим як перейти до детального огляду існуючих систем, важливо відзначити, що на ринку представлено значну кількість програмних рішень (VMS та NVR), спрямованих на організацію відеоспостереження та відеоаналітики. Кожна з цих систем має свої переваги та недоліки і спрямована на вирішення певного спектра завдань. Під час аналізу існуючих рішень буде здійснено порівняння їхніх характеристик, з'ясовано основні обмеження та переваги. Це дозволить обґрунтувати необхідність розробки власної системи, яка максимально відповідатиме сучасним вимогам користувачів та бізнесу щодо швидкодії, точності детекції об'єктів та гнучкості інтеграції.

Blue Iris. Це професійне програмне забезпечення для відеоспостереження, призначене для роботи виключно на базі OS Windows. Позиціонується як комплексне рішення для запису, управління та аналізу відеопотоків з IP-камер.

Особливості:

- програмне забезпечення для відеоспостереження, що підтримує до 128 камер.
- має вбудований веб-сервер UI3, підтримує безперервний запис або запис за датчиками руху/звуку.

Переваги:

- вбудована підтримка штучного інтелекту (ШІ) без потреби у зовнішньому ПЗ.
- потужні налаштування сповіщень та інтеграція зі сторонніми ШІ-серверами (наприклад, CodeProject AI).
- можливість запуску як фонові служби Windows.

Недоліки:

- високі вимоги до апаратного забезпечення (сучасні процесори Intel, від 8 ГБ ОЗП).
- платні ліцензії (\$39.95 за 1 камеру, \$99.95 за повну версію), ліцензійні ключі вважаються такими, що не підлягають поверненню.

					ВКРБ-123.26.0010.00.00.ПЗ	Арк
Вим	Арк.	№ докум.	Підп.	Дата		10

– Технічна підтримка телефоном або через віддалений робочий стіл дуже обмежена.

Підтримувані платформи. Працює виключно на Microsoft Windows 10/11 (64-біт) або відповідному сервері. Також передбачено мобільний доступ.

Scripted. Це насамперед програмне забезпечення, написане на Node.js, що працює як швидкий відеоміст. Основна ідея цього проєкту - надати користувачам функціонал не відмінний від професійного, але в зручному інтерфейсі з особливим акцентом на Google Home, Apple HomeKit Secure Video та Amazon Alexa.

Особливості:

– платформа з відкритим вихідним кодом (Open Source) для інтеграції відео, яка отримує потоки з камер та передає їх до платформ розумного дому (Apple HomeKit, Google Home, Alexa, Home Assistant).

Переваги:

– наявність плагіна Scripted NVR, який забезпечує цілодобовий запис (24/7), розумні виявлення та має власні десктопні й мобільні додатки.

– підтримує інтеграцію з великими мовними моделями (LLM) та вбудований ШІ камер.

Недоліки:

– залежність від налаштування різних плагінів для повноцінної роботи.

Підтримувані платформи. Дуже висока кросплатформність. Можна встановити на macOS, Windows (через Desktop або PowerShell), Linux (Docker), Proxmox VE, Unraid.

					ВКРБ-123.26.0010.00.00.ПЗ	Арк
						11
Вим	Арк.	№ докум.	Підп.	Дата		

NX Witness. Це високоефективне професійне програмне забезпечення (VMS) для управління відео, що входить до екосистеми Powered by Nx. Ця система відома своєю винятковою швидкістю роботи, легкістю клієнтського ПЗ та відкритою архітектурою для масштабних інтеграцій.

Особливості:

– Уніфікована екосистема для відеоспостереження, що об'єднує компоненти Server, Cloud, Desktop та Mobile.

Переваги:

– Універсальна сумісність - підтримує понад 45 000 пристроїв від більш ніж 790 виробників. Гнучке управління даними (локальні, хмарні або гібридні рішення) з функціями відмовостійкості для надійності. Наявність набору інструментів розробника (Nx Developer Toolkit).

Недоліки:

– Висока вартість ліцензування каналів.
– Обмежена робота зі складними сценаріями автоматизації без програмування.
– Перехід на модель підписки для корпоративних функцій.

Підтримувані платформи. Кросплатформне рішення. Управління можливе через десктопну версію (Nx Desktop), хмару (Nx Cloud) для віддаленого доступу та мобільні додатки для iOS та Android (Nx Mobile)

Synology Surveillance Station. це професійна система управління відео (VMS), яка постачається як безкоштовний додаток до мережевих сховищ (NAS) компанії Synology. Проєкт орієнтований на максимальну стабільність та простоту використання за принципом «все в одному».

Особливості:

– Фірмове програмне забезпечення для відеоспостереження від компанії Synology.

					ВКРБ-123.26.0010.00.00.ПЗ	Арк
						12
Вим	Арк.	№ докум.	Підп.	Дата		

Переваги:

– Глибока інтеграція з апаратними рішеннями Synology. Підтримує широкий парк пристроїв виробника, включаючи спеціалізовані сервери відеоаналітики DVA (DVA1622, DVA3221), мережеві реєстратори NVR, а також системи DiskStation і RackStation. Це робить систему вкрай стабільною та оптимізованою під конкретне залізо.

Недоліки:

– Апаратна прив'язка. Система вимагає купівлі обладнання від компанії Synology.

Підтримувані платформи. Працює виключно локально як частина операційної системи DiskStation Manager (DSM) на пристроях Synology (NAS/NVR)

Отже, виділивши всі ключові особливості кожного аналога Frigate NVR можна тепер навести таблицю порівняння між ними. Основними пунктами по оцінці будуть такі критерії: ліцензія, основна операційна система, детекція штучного інтелекту, апаратне прискорення, інтеграція з Internet of Things(IoT), вимоги до заліза.

Як видно з порівняльної таблиці (див. Таблиця 2.1) Frigate NVR досить непогане рішення, так як не відстає від своїх аналогів, які переважно комерційні. Але має і свої недоліки - він не модульний, що можна дуже легко поправити якщо створити спеціалізований фреймворк, або ж програмне забезпечення з такою можливістю. Це йому не заважає виконувати мінімальний набір функцій для бізнес задач, та зважаючи, що його мета це детекція та мінімізація помилок - це навіть найкраще рішення якщо використовувати його для створення кастомних рішень.

Вим	Арк.	№ докум.	Підп.	Дата

Таблиця 2.1 — Порівняння Frigate NVR та його аналогів за основними ознаками

Критерії порівняння	Frigate NVR	Blue Iris	Scrypted	NX Witness	Synology Surveillance Station
Ліцензія	Open Source	Комерційна	Гібридна	Комерційна	Комерційна
Основна ОС	Linux (Docker)	Windows	Docker / Linux / Mac	Linux / Windows	DSM (Власна OS Synology)
Детекція ІІІ	Вбудована (YOLO, TensorFlow)	Зовнішня (CodeProject ,AI)	Модульна	SDK та плагіни	Вбудована (Deep Video Analytics)
Апаратне прискорення	Coral TPU, GPU, OpenVINO	QuicSync, NVDEC	TensorRT, OpenVino, Coral	GPU	Спеціалізовані чипи, серія DVA
Інтеграція IoT	MQTT	MQTT та HTTP	HomeKit, Google	REST API	Webhooks та Action Rules
Вимоги до заліза	Низькі вимоги; при наявності TPU	Високі вимоги; потужні CPU і GPU	Дуже низькі вимоги, через модульність	Низькі вимоги; ефективний клієнт	Залежить від NAS

Хоча у порівнянні і зазначав про відсутність модульності, з інженерної точки зору це компенсується відкритістю вихідного коду. Це дає можливість розробнику впроваджувати кастомні моделі детекції, навчені під специфіку бізнесу, а не створювати його з конструктору плагінів або іншими засобами.

Так як доступ до Frigate ми отримаємо через його API, то це не заважає створювати власні логічні шари такі як: обробка подій, що використовує API

систему. Що в свою чергу не обмежується тільки відео, а і доступ до ауді, тобто повна інтелектуальна система.

Також відкривається максимальна свобода в тонкому налаштуванні параметрів детектора під конкретні архітектури процесорів, що може призвести до оптимізації використання пам'ять.

Таким чином, Frigate NVR є оптимальним фундаментом для кваліфікаційної роботи. Він поєднує в собі і потужність сучасних нейромереж, що до речі дуже стрімко розвиваються, і дозволяє гнучко їх перепідключати, завдяки гнучкій системі відкритого вихідного коду. Це є ідеальними умовами для реалізації передової, надійної системи відеоспостереження, яка відповідає сучасним вимогам безпеки та аналізу даних за допомогою ШІ.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Інтелектуальну систему відеоспостереження, яка розробляється в рамках даної дипломної роботи, обрано розробляти на мові програмування Python 3.14.4 (для серверної та аналітичної частини) та фронтенд на React 19.1.1 + Electron 42.0.1 (для клієнтської частини). Цей вибір обґрунтовується рядом переваг та характеристик, які забезпечують ефективність, зручність та швидкість розробки, а також високу продуктивність при роботі зі штучним інтелектом.

Python 3.14.4

Четвертий стабільний реліз з обслуговування (maintenance release) мажорної версії 3.14. Він включає близько 337 виправлень помилок, покращень процесу збірки, оптимізацій C-API та оновлень документації, що накопичилися з часів версії 3.14.3.

Нові функції:

- PEP 779 : Вільний потоковий Python офіційно підтримується.

					ВКРБ-123.26.0010.00.00.ПЗ	Арк
Вим	Арк.	№ докум.	Підп.	Дата		15

- PEP 649 : Оцінювання анотацій тепер відкладається, що покращує семантику їх використання.
- PEP 750 : Шаблонні рядкові літерали (t-рядки) для обробки користувацьких рядків з використанням знайомого синтаксису f-рядків.
- PEP 734 : Кілька інтерпретаторів у stdlib.
- PEP 784 : Новий модуль compression.zstd, що забезпечує підтримку алгоритму стиснення Zstandard.
- PEP 758 : вирази ехсептта ехсепт*тепер можуть не ставити дужки.
- Підсвічування синтаксису в PyREPL та підтримка кольору в інтерфейсах командного рядка unittest , argparse , json та calendar .
- PEP 768 : Інтерфейс зовнішнього налагоджувача без накладних витрат для CPython.
- модуль тепер підтримує UUID версій 6-8uuid , а генерація версій 3-5 відбувається до 40% швидше.
- PEP 765 : Заборонити return/ break/ continueщо виходять з finallyблоку.
- PEP 741 : Покращений C API для налаштування Python.
- новий тип інтерпретатора . Для деяких новіших компіляторів цей інтерпретатор забезпечує значно кращу продуктивність. Поки що здійснюйте підписку, потрібна збірка з вихідного коду.
- покращені повідомлення про помилки.
- вбудована реалізація HMAC з формально перевіреним кодом з проекту HACSL.
- новий інтерфейс командного рядка для перевірки запущених процесів Python за допомогою асинхронних завдань.
- модуль pdb тепер підтримує віддалене підключення до запущеного процесу Python .

Стабілізація Free-threaded CPython (режим без GIL)

					ВКРБ-123.26.0010.00.00.ПЗ	Арк
Вим	Арк.	№ докум.	Підп.	Дата		16

Найбільший прорив мови. Python 3.14.4 пропонує високостабільну роботу без глобального блокування інтерпретатора (Global Interpreter Lock). Це означає, що програма може повноцінно і паралельно використовувати всі доступні ядра багатоядерних процесорів для обчислень.

Інтеграція JIT-компілятора (Just-In-Time)

Повноцінне впровадження компілятора типу «copy-and-patch». Замість стандартної інтерпретації, Python тепер компілює гарячі ділянки коду (математичні операції, складні цикли) у швидкий машинний код «на льоту», забезпечуючи значний приріст базової продуктивності.

Відкладена оцінка анотацій типів (Type Hints)

Завдяки новому механізму, складні підказки типів більше не гальмують запуск програми. Вони аналізуються лише тоді, коли до них іде фактичне звернення (під час дебагу або роботи інструментів статичного аналізу). Це різко прискорює старт мікросервісів (наприклад, на базі FastAPI).

Менеджер встановлення Python

- Новий менеджер інсталяції, який можна встановити з Магазину Windows або зі сторінки завантаження.
- Традиційний інсталятор залишатиметься доступним у випусках 3.14 та 3.15.

Беручи до уваги вище написане, та основні властивості Python як універсальність, швидкість реалізації ідей, гігантську спільноту, ось чому я обираю саме цю мову програмування для мого проєкту випускної кваліфікаційної роботи бакалавра.

React 19.1.1

React – відкрита JavaScript бібліотека, випущена 2013 році, для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту вебсторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Meta і спільнотою індивідуальних розробників. Стабільний випуск 19.1.1 від 28 липня 2025.

Загальні зміни:

- спрощена робота з ref
- оновлений Context API
- вбудована підтримка метаданих (Document Metadata)
- нативне управління ресурсами (Asset Loading)

Компоненти сервера

Серверні компоненти – це нова опція, яка дозволяє рендерити компоненти заздалегідь, перед їх об'єднанням, у середовищі, окремому від вашої клієнтської програми або SSR-сервера. Це окреме середовище є «сервером» у React Server Components. Серверні компоненти можуть запускатися один раз під час збірки на вашому CI-сервері або для кожного запиту за допомогою веб-сервера.

React 19 включає всі функції React Server Components, включені з каналу Canary. Це означає, що бібліотеки, що постачаються з Server Components, тепер можуть орієнтуватися на React 19 як на залежність однорангового вузла з react-server умовою експорту для використання у фреймворках, що підтримують повнорозмірну архітектуру React.

Actions та нативна асинхронність

Введено концепцію «Actions» для автоматичного управління асинхронними процесами. Нові хуки, такі як `useActionState` та `useFormStatus`, автоматично обробляють стани завантаження (pending), успішного виконання та помилок, позбавляючи розробника необхідності вручну створювати змінні `isLoading` чи `isError`.

Універсальний хук use()

Інструмент, який дозволяє зчитувати значення ресурсів (наприклад, результати Promise або дані з Context) безпосередньо всередині функції рендерингу, навіть усередині умов (if/else), що раніше було суворо заборонено правилами хуків React.

Electron 42.0.1

Electron - це потужний фреймворк із відкритим вихідним кодом, випущений 15 липня 2013, остання версія якого дозволяє створювати кросплатформні десктопні застосунки (для Windows, macOS, Linux) з використанням веб-технологій (HTML, CSS, JavaScript/React).

Новітні рушії Chromium та V8

Версія 42.0.1 базується на найсучасніших збірках Chromium, що забезпечує підтримку останніх веб-стандартів та максимальну швидкодію виконання JavaScript-коду.

Посилена безпека (Context Isolation та Sandboxing)

У нових версіях жорстко розмежовано доступ між веб-інтерфейсом (Renderer Process) та системним рівнем (Main Process). Це унеможливорює виконання шкідливого коду навіть у разі компрометації фронтенду.

Високоєфективний IPC (Inter-Process Communication)

Оптимізовано протоколи обміну повідомленнями між процесами додатку, що дозволяє передавати великі масиви даних з мінімальними затримками.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускню кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи відеоспостереження на основі технології Frigate NRV.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

- 1) Проаналізувати аналогічні системи, зазначити їх переваги та недоліки;
- 2) Розробити серверну частину програмного забезпечення, що забезпечує зв'язок між всіма модулями;
- 3) Організувати зберігання даних;
- 4) Інтегрувати модуль Frigate NRV та III модуль до програмного забезпечення.
- 5) Розробити алгоритми для обробки об'єктів - їх повторної ідентифікації.
- 6) Розробити клієнтську частину програмного забезпечення, що забезпечує керування системою;
- 7) розробити методики, які забезпечать впровадження системи в промислову експлуатацію;
- 8) Сформулювати висновки, навести одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Для забезпечення кросплатформенності, ізоляції процесів та зручності розгортання на кінцевих серверах (Edge-пристроях), усю кодову базу та інфраструктурні компоненти я реалізував у вигляді мікросервісної архітектури на базі Docker. Використання технології контейнеризації дозволяє уникнути проблеми «на моєму комп'ютері це працює» та гарантує ідентичність середовищ розробки і продакшену.

Оркестрація контейнерів здійснюється за допомогою docker-compose. Інфраструктурний стек складається з наступних взаємопов'язаних сервісів, об'єднаних у віртуальну мережу (bridge network):

- 1. **Frigate NVR:** Ядро системи відеоаналітики, яке має прямий доступ до апаратних прискорювачів (Coral TPU або GPU) через прокидання пристроїв (device mapping) у контейнер.
- 2. **MQTT Broker (Eclipse Mosquitto):** Відповідає за швидку та асинхронну маршрутизацію повідомлень про події між Frigate та моїм бекендом.
- 3. **TimescaleDB (PostgreSQL):** Сховище структурованих подій та метаданих. Контейнер налаштовано з оптимізованими параметрами пам'яті для роботи з гіпертаблицями.
- 4. **ChromaDB:** Векторна база даних для збереження ембеддінгів (ознак об'єктів), необхідних для функції Re-ID.
- 5. **Backend API (FastAPI):** Логічний центр системи, написаний на Python, який агрегує дані та взаємодіє з LLM.
- 6. **Frontend (React/Vite):** Веб-сервер, що віддає клієнтську частину додатку.

7. **Ollama:** Локальний сервер для запуску квантованих великих мовних моделей, який обробляє запити на генерацію «Дедуктивної історії».

Функціонування системи базується на принципах периферійних обчислень (Edge Computing), де вся обробка відеоаналітики відбувається локально на об'єкті без використання хмарних сервісів. Робота системи поділяється на чотири основні етапи.

Перший етап – захоплення та детекція

1. **Захоплення потоку:** система підключається до IP-камер через протоколи RTSP/ONVIF. За допомогою бібліотеки FFmpeg відеопотік розділяється на два субпотіки: один для запису (висока якість) та один для детекції (низька роздільна здатність для економії ресурсів).

2. **Інтелектуальна детекція:** Кожен кадр аналізується нейромережею YOLO. Для забезпечення швидкодії «тут і зараз» використовуються апаратні прискорювачі, такі як Google Coral TPU або NVIDIA GPU (TensorRT), що дозволяє розвантажити центральний процесор.

3. **Фільтрація подій:** за допомогою налаштованих «масок» та «зон» система ігнорує статичні об'єкти та фокусується лише на значущих подіях.

Другий етап – передача та збереження даних

1. **Шина повідомлень:** Як тільки об'єкт детектується, повідомлення про подію надсилається через брокер Mosquitto (MQTT).

2. **База даних метаданих:** Розроблений на Python (FastAPI) бекенд отримує дані з MQTT та записує їх у PostgreSQL із розширенням TimescaleDB. Це дозволяє ефективно зберігати інформацію про тип об'єкта, час його появи та тривалість перебування в зоні моніторингу.

3. **Зберігання медіа:** Відеофрагменти записуються на диск лише у разі підтвердженої детекції «живого» об'єкта, що дозволяє значно економити місце на накопичувачах. Звичайний запис відео стискається та архівується в кінці дня.

Третій етап – формування «Дедуктивної історії» об’єкта

- 1. Векторизація та пошук:** для кожного детектованого об’єкта створюється векторне представлення, яке зберігається у ChromaDB. Це дозволяє ідентифікувати одного і того ж об’єкта на різних камерах протягом дня.
- 2. Контекстний аналіз:** Система аналізує не лише цільовий об’єкт, а й оточення: які інші об’єкти знаходилися поруч у той самий момент часу.
- 3. Локальний ШІ (LLM):** Зібрані метадані за день передаються до локальної мовної моделі Оllama. Модель синтезує розрізнені логи у зрозумілу текстову історію.

Четвертий етап – аналітичний

- 1. Оперативна аналітика:** моніторинг в реальному часі через веб-інтерфейс.
- 2. Ретроспективний аналіз:** Генерація щоденних підсумків, узагальнення щоденне, тижневе або інший проміжок часу.

У продовження опису мого проєкту, я деталізую третій етап - «Дедуктивну історію», оскільки саме він перетворює мою систему зі звичайного засобу фіксації на інтелектуальний інструмент розслідування. Для мене цей етап є найбільш значущим, адже тут я поєднаю можливості векторної математики та генеративного ШІ для відновлення логічного ланцюжка подій.

Мій підхід до дедукції базуватиметься на трьох «китах»: *ідентифікації, контекстуалізації та інтерпретації*.

Векторна ідентифікація та Re-ID

Для реалізації функції «Дедуктивної історії» необхідно вирішити складну задачу комп’ютерного зору, Re-Identification (Re-ID), тобто розпізнавання одного й того ж об’єкта на різних камерах або в різний час за умови зміни ракурсу чи освітлення. Для цього застосовується механізм екстракції ознак (Feature Extraction).

Кожен детектований системою Frigate NVR об'єкт проходить через додаткову згорткову нейромережу, яка відкидає просторову інформацію, залишаючи лише семантичні ознаки (наприклад, колір одягу, загальні пропорції, текстури).

Ці ознаки формують n -мірний вектор $V = [v_1, v_2, \dots, v_n]$, який називається векторним представленням або ембедінгом (embedding). У розроблюваній системі для збереження та швидкого пошуку цих багатовимірних векторів розгорнуто спеціалізовану базу даних ChromaDB.

Для визначення того, чи належать два вектори (отримані з різних камер) одному об'єкту, система обчислює метрику косинусної подібності (Cosine Similarity). На відміну від звичайної евклідової відстані, косинусна подібність вимірює кут між двома векторами, що робить її стійкою до змін у масштабі об'єкта. Формула (3.1) косинусної подібності для векторів A та B має вигляд:

$$\cos(\theta) = \frac{A \cdot B}{|A| |B|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}} \quad (3.1)$$

де $\cos(\theta)$ - значення косинусної подібності;

A, B - вектори ознак об'єктів;

n - розмірність векторного простору;

A_i, B_i - компоненти векторів ознак.

Хронологічна та просторова кореляція

Після ідентифікації об'єкта я звернуся до PostgreSQL (TimescaleDB), щоб витягнути всі часові мітки та зони його появи. Тоді згрупую ці дані в єдиний часовий ряд. Система аналізуватиме паралельні події: хто був поруч, які предмети переміщувалися та чи змінювався стан контрольних зон.

III-синтез «Дедуктивної історії»

Фінальним і найцікавішим кроком я зроблю передачу зібраного масиву метаданих до локальної моделі Ollama (Llama 3). Замість того, щоб змушувати користувача читати таблиці логів, я змушу III написати зв'язний звіт.

Далі перейдемо до тонкого налаштуванню LLM.

Я спроектую систему так, щоб користувачеві не потрібно було володіти навичками промпт-інжинірингу. Замість введення складних інструкцій, оператор просто обиратиме конкретний об'єкт (наприклад, клікнувши по ньому у списку детекцій) і вводитиме максимально просте запитання.

Фінальним етапом аналітичного контуру системи є логів подій з бази даних на зрозумілий текстовий звіт. Для дотримання концепції Edge Computing та забезпечення повної конфіденційності у системі розгорнуто локальну велику мовну модель за допомогою фреймворку Ollama. Для стабільної роботи на локальному обладнанні використовується модель Llama 3, квантована до 4-бітних цілих чисел (INT4).

Бекенд, написаний на FastAPI, виконує роль прошарку між агрегованими даними (PostgreSQL/TimescaleDB) та локальною LLM. Коли оператор ініціює запит щодо конкретного об'єкта, бекенд формує жорстко структурований системний промпт (System Prompt). Цей промпт обмежує свободу генерації моделі, мінімізуючи ризик «галюцинацій», і змушує її працювати виключно з переданим масивом фактів.

Структура автоматизованого промпту включає три блоки:

- 1. Системна роль: «Ти є суворим аналітиком систем безпеки. Твоє завдання - написати лаконічний хронологічний звіт про переміщення об'єкта без використання емоційних або художніх описів».
- 2. Обмеження: «Спирайся виключно на наданий масив даних (JSON). Якщо інформації бракує, констатуй цей факт».

3. **Контекст даних:** Сюди ін'єктується зріз бази даних, попередньо відфільтрований за унікальним ідентифікатором об'єкта (встановленим на основі косинусної подібності векторів).

Приклад фрагмента масиву даних, який бекенд передає моделі:

```
[ {"timestamp": "10:15", "camera": "Cam_1", "zone": "Entrance",  
  "object_id": "452", "confidence": 0.92},  
  {"timestamp": "10:22", "camera": "Cam_4", "zone": "Section_B",  
  "object_id": "452", "confidence": 0.88} ]
```

Приклад коректної взаємодії:

Запит користувача: «Опиши, що сталося з Об'єктом #452 впродовж дня».

Внутрішня механік: Мій бекенд на FastAPI автоматично сформує розширений контекст для LLM. Він витягне з PostgreSQL всі мітки часу, назви зон та метадані про наявність додаткових предметів, а з ChromaDB - підтверджені зв'язки цього об'єкта між камерами.

Код для налаштування ШІ:

```
.....  
system_instruction = (  
    "Ти - інтелектуальний помічник системи Omniscience NVR. "  
    "Твоє завдання: аналізувати логи спостереження та файлову  
структуру архіву. "  
    "СПІЛКУЙСЯ ТІЛЬКИ УКРАЇНСЬКОЮ МОВОЮ. Будь лаконічним, але  
точним."  
)  
payload = {  
    "model": config.AI_MODEL,  
    "prompt": f"{system_instruction}\nКонтекст системи:  
{context}\nПитання користувача: {prompt}",  
    "stream": False  
}  
.....
```

Результат роботи ШІ:

«Об'єкт #452 (людина) з'явився на Камері №1 о 10:15 із чорною сумкою. Протягом 5 хвилин перебував у зоні поліції №3. О 10:22 об'єкт був зафіксований Камерою №4 у Секції Б, проте сумка в руках уже була відсутня. Ймовірне місце залишення речей: Секція А, район поліції №3».

3.2 Розробка структурної схеми

Розглянемо розроблену у бакалаврському проекті структурну схему роботу системи (див. Рисунок 3.1). Програма призначення для системи відеоспостереження на основі технології Frigate NVR.

Призначення розробленого програмного забезпечення, реалізація системи відеоспостереження на основі Frigate NVR. Необхідне для виявлення об’єктів під час відеоспостереження і реагувати на події, що вони ініціюють, або приймають участь. Має наступні можливості:

- детекція об’єктів;
- детекція об’єктів за класифікацією;
- детекція руху;
- детекція шляху;
- первинна ідентифікація об’єкта;
- повторна ідентифікація об’єкта;
- трансляція відеопотоку з камер відеоспостереження у клієнт;

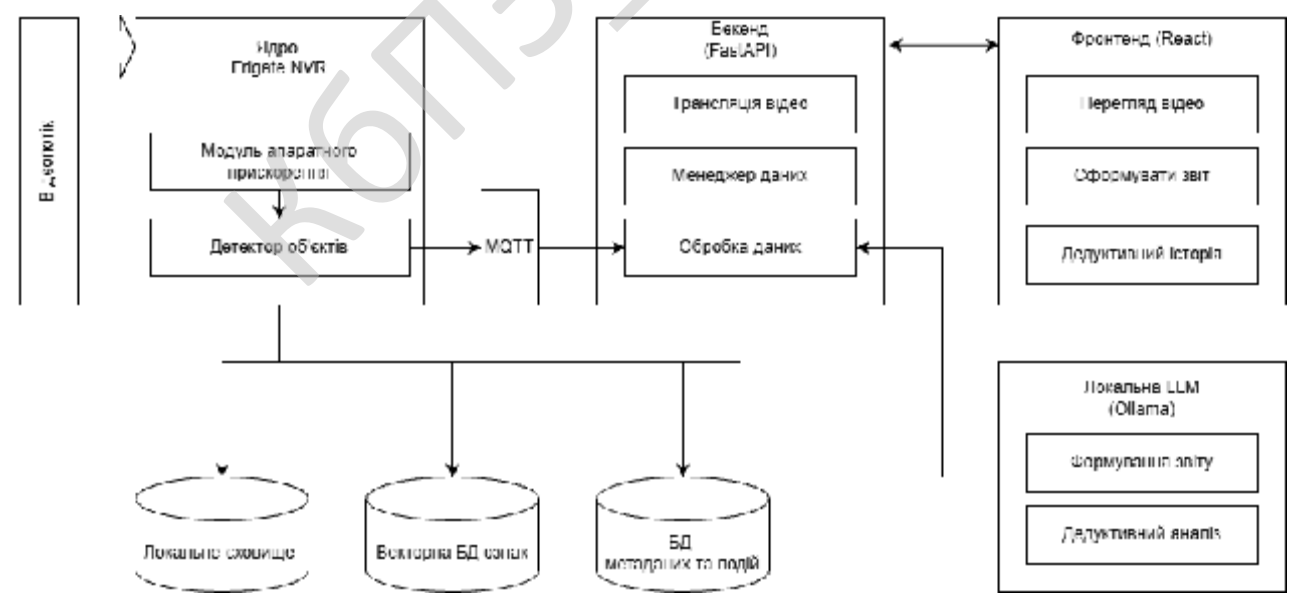


Рисунок 3.1 — Структурна схема системи відеоспостереження на основі Frigate NVR

На структурній схемі зображено чотири основні логічні блоки, три типи баз даних, вхідний відеопотік та шину обміну даними (брокер MQTT). Кожен із цих елементів відіграє критичну роль у забезпеченні швидкодії та надійності системи.

Підсистема збору відеоданих (Відеопотік)

Вхідною точкою системи є відеопотік, що надходить від IP-камер по протоколу RTSP (Real Time Streaming Protocol). Для оптимізації роботи система працює з двома типами потоків від кожної камери: субпоток (низька роздільна здатність, наприклад 720p) для швидкої програмної детекції, та основним потоком (висока роздільна здатність, 4K/1080p) для збереження на жорсткий диск та перегляду користувачем. Це дозволяє уникнути перевантаження центрального процесора при розкодуванні відео.

Ядро відеоаналітики (Frigate NVR)

Цей блок є першим рівнем інтелектуальної обробки. Ядро Frigate NVR безперервно аналізує субпотік. Робота цього модуля поділяється на два етапи:

- легковагова детекція руху на рівні пікселів.
- при виявленні руху кадр передається на модуль апаратного прискорення (Google Coral TPU або графічний процесор NVIDIA).

Завдяки використанню апаратного тензорного прискорювача, система здатна виконувати передбачення нейромережі за лічені мілісекунди. Якщо в кадрі (наприклад, у кімнаті чи на периметрі) з'являється людина або інший класифікований об'єкт, Frigate миттєво фіксує його координати (bounding boxes). Оброблений потік записується на диск у Локальне сховище (файлова система) у вигляді сегментованих відеофайлів.

Комунікаційний рівень (Брокер MQTT)

Для забезпечення асинхронності та масштабованості архітектури, зв'язок між ядром аналітики та логічним сервером реалізовано через брокер MQTT. При успішній детекції об'єкта, Frigate формує JSON-повідомлення, що містить тип об'єкта, рівень впевненості моделі, координати та ідентифікатор камери,

після чого публікує його в певний топик (наприклад, frigate/events). Брокер MQTT забезпечує миттєву маршрутизацію цього повідомлення всім підписникам.

Серверна логіка (Бекенд на FastAPI)

Бекенд - це центральний мозковий центр розробленої системи, написаний на мові Python з використанням фреймворку FastAPI. Цей модуль постійно «слухає» топіки брокера MQTT. Отримавши сповіщення про новий об'єкт, бекенд ініціює наступний ланцюжок дій:

- запитує у Frigate кращий кадр (snapshot/crop) зафіксованого об'єкта.
- обробляє зображення та перетворює його у векторне представлення (embedding) за допомогою спеціалізованих алгоритмів.

- виконує маршрутизацію збереження даних у дві спеціалізовані бази.

Гібридна підсистема зберігання даних

Система використовує три різні сховища, кожне з яких оптимізоване під свій тип даних:

- **Локальне сховище файлів:** Використовується ядром Frigate для збереження важких MP4-файлів відеоархіву.

- **База даних часових рядів:** Зберігає всі метадані події з жорсткою прив'язкою до часу. Тут записується: яка камера спрацювала, коли почалася подія, коли завершилася, та який тип об'єкта виявлено. Це необхідно для побудови графіків та фільтрації відео на таймлайні клієнта.

- **Векторна база даних:** Спеціалізована база для збереження векторів ознак об'єктів. Коли бекенд зберігає сюди новий вектор, він виконує математичний пошук на основі косинусної подібності. Якщо вектор співпадає з існуючими в базі (з високим ступенем вірогідності) відбувається ре-ідентифікація. Об'єкту присвоюється унікальний глобальний ідентифікатор, який діє не в межах однієї камери, а в масштабах всієї мережі відеоспостереження на тривалий термін (відповідно до конфігурації).

Вим	Арк.	№ докум.	Підп.	Дата

ВКРБ-123.26.0010.00.00.ПЗ

Арк

29

Модуль клієнта відповідає за візуальне представлення даних та взаємодію з користувачем. Завдяки архітектурі системи, бекенд передає клієнту як прямий відеопотік, так і субпотік, що дозволяє користувачеві відслідковувати об'єкти безпосередньо в інтерфейсі (включаючи відображення рамок детекції). Робота з архівом може відбуватися у двох режимах:

- **Пасивний:** перегляд трансляції у вигляді матриці камер.
- **Активний:** взаємодія з таймлайном, де всі детекції вже відмічені маркерами завдяки інформації з бази даних часових рядів.

Крім того, клієнт є ініціатором складних аналітичних запитів до бекенду, таких як формування звітів або запуск дедуктивного аналізу.

Модуль інтелектуального аналізу (Локальна LLM)

Найвищим рівнем розробленої системи є модуль локальної великої мовної моделі (LLM, наприклад, сімейства Llama або Mistral, розгорнутої через Ollama). Цей модуль підключається до роботи лише за ініціативою клієнта і виконує дві надскладні функції, що перетворюють розрізнені дані на структуровану інформацію:

Формування звітів: При ініціації запиту на звіт (наприклад, за день, тиждень або місяць), LLM отримує від бекенду текстові метадані про всі події з реляційної бази даних (без необхідності «дивитися» саме відео). Модель аналізує статистику (кількість людей, періоди пікової активності, маршрути) і генерує легкочитний текстовий звіт для адміністратора безпеки.

Дедуктивний аналіз поведінки: Це унікальна функція системи. Коли користувач обирає конкретний об'єкт (користуючись результатами векторної ре-ідентифікації), бекенд формує хронологічний ланцюжок усіх появ цієї людини на різних камерах об'єкта. Цей структурований масив (наприклад: "Об'єкт X увійшов через головний вхід о 12:00, був зафіксований біля серверної о 12:05, покинув будівлю через чорний хід о 12:20") передається до LLM у вигляді спеціального промпту. Мовна модель аналізує історію переміщень, виявляє

аномалії, генерує контекстну історію об'єкта та робить дедуктивний висновок щодо логіки його дій, що є безцінним інструментом при розслідуванні інцидентів.

Життєвий цикл події в системі

Для кращого розуміння роботи структурної схеми, доцільно розглянути життєвий цикл проходження даних на прикладі появи на об'єкті нової особи:

1. Камера фіксує рух та відправляє відеопотік по RTSP до Frigate NVR.
2. Ядро Frigate делегує кадр тензорному прискорювачу, який розпізнає на відео людину.
3. Frigate створює подію та відправляє сповіщення через брокер MQTT, паралельно зберігаючи фрагмент відео на локальний диск.
4. Бекенд отримує сповіщення, завантажує кадр з обличчям/силуетом людини та формує векторне представлення.
5. У Векторній БД відбувається перевірка: чи зустрічалася ця людина раніше. Якщо ні, створюється новий профіль глобального об'єкта, а час і координати записуються у БД часових рядів.
6. Клієнт миттєво отримує оновлення інтерфейсу через WebSocket і відображає подію в стрічці.
7. Оператор системи, помітивши підозрілу особу, натискає кнопку «Дедуктивний аналіз», після чого Локальна LLM збирає всі метадані щодо цієї особи та видає звіт про її активність.

Таким чином, розроблена структурна схема забезпечує чіткий розподіл обов'язків (мікросервісна архітектура), що гарантує високу продуктивність, масштабованість та відмовостійкість системи відеоспостереження на всіх етапах її роботи.

3.3 Розробка функціональної схеми

Представлена функціональна схема (див. Рисунок 3.2) відображає послідовний процес повної трансформації вхідного RTSP-відеопотоку в аналітичні дані інтелектуальної системи.

Блок попередньої обробки відео. Відповідає за первинну взаємодію з IP-камерами та підготовку даних для подальшого аналізу. Він включає наступні функції:

- захоплення основного потоку - отримання відео високої роздільної здатності для його подальшого збереження в архів;
- захоплення суб-потoku - отримання відео меншої роздільної здатності для швидкого аналізу з мінімальним навантаженням на систему;
- декодування та нормалізація - розпакування кадрів відеопотоку та приведення їх до єдиного формату й розміру;
- апаратне прискорення декодування - делегування процесів обробки відео графічному процесору (GPU) для розвантаження центрального процесора;
- піксельна детекція руху - швидке виявлення динамічних змін у кадрі для активації процесу розпізнавання лише за реальної потреби;
- кешування відеопотоку - тимчасове утримання останніх секунд трансляції в пам'яті для збереження кліпу за кілька секунд до початку події (pre-capture).

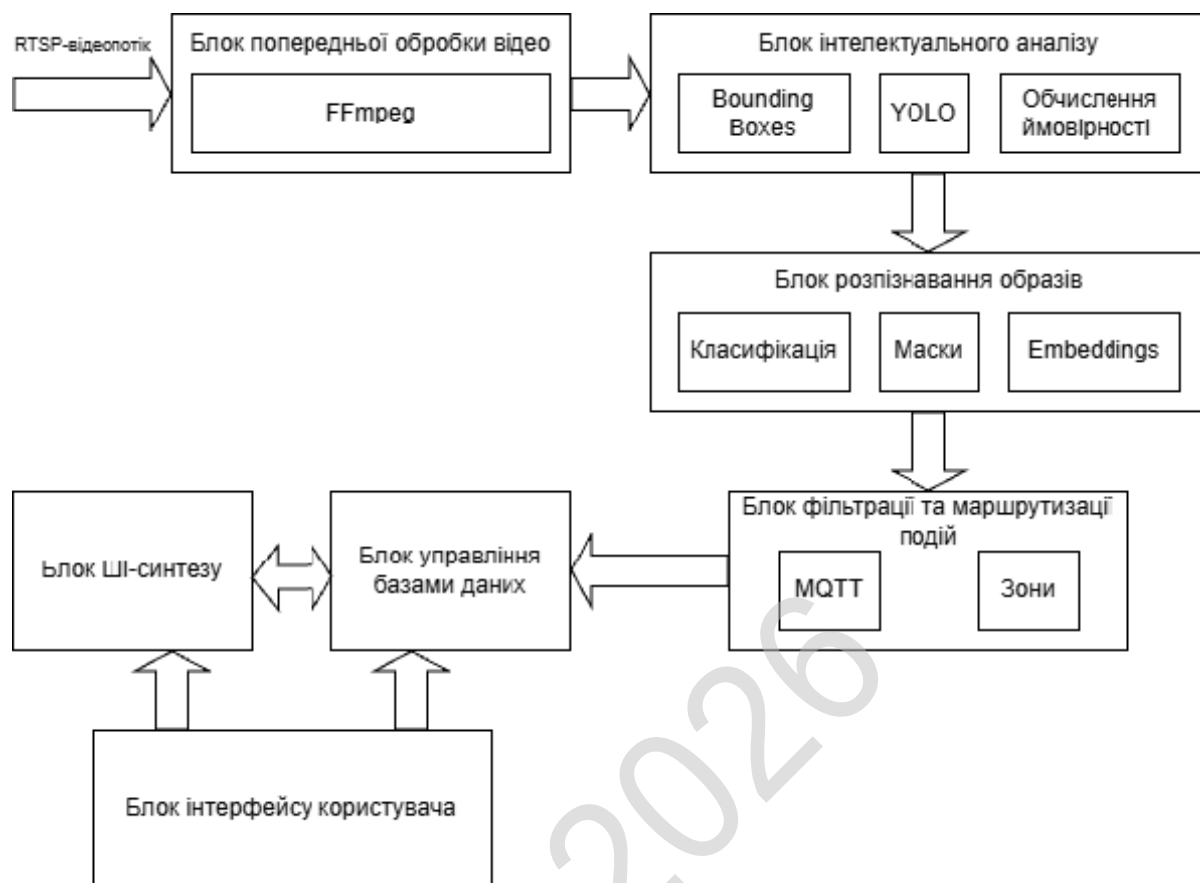


Рисунок 3.2 — Функціональна схема

Блок розпізнавання образів. Є ключовим компонентом комп'ютерного зору (на базі ядра Frigate та апаратних прискорювачів). Виконує такі функції:

- висновок нейромережі - швидке виконання математичних розрахунків моделі машинного навчання на підготовленому кадрі;
- класифікація - визначення типу виявленого об'єкта (наприклад, людина, легковий автомобіль, тварина);
- локалізація об'єктів - визначення точних координат знайденого об'єкта на площині кадру;
- відстеження об'єктів у межах кадру - утримання фокуса на рухомому об'єкті (локальний трекінг) для запобігання втраті цілі при короткочасному перекритті;

- оцінка якості кадру - програмне відкидання розмитих або засвічених кадрів перед їх глибинним аналізом.

Блок фільтрації та маршрутизації подій. Відповідає за комунікацію між мікросервісами та відсіювання інформаційного «сміття». Виконує такі функції:

- просторова фільтрація - ігнорування детекцій у статичних зонах-масках та врахування подій лише у визначених «зонах інтересу»;
- формування подій - створення структурованих повідомлень (наприклад, у форматі JSON) із повною інформацією про виявлений об'єкт;
- маршрутизація даних - передача сформованих повідомлень від ядра аналітики до бекенду через шину даних (брокер MQTT);
- дедуплікація подій - об'єднання серії швидких однотипних подій в одну логічну сесію для уникнення дублювання записів у базі;
- пріоритезація сповіщень - визначення рівня важливості події для її першочергової відправки клієнту.

Блок інтелектуального аналізування. Відповідає за глибинну обробку даних на стороні логічного бекенду (FastAPI). Виконує такі функції:

- обробка обмежувальних рамок (Bounding Boxes) - динамічне вирізання силуету об'єкта з кадру та аналіз перетину координат із забороненими зонами;
- екстракція ознак (Feature Extraction) - перетворення візуального образу об'єкта на унікальний математичний вектор (embedding);
- повторна ідентифікація (Re-Identification) - розпізнавання раніше зафіксованих об'єктів на інших камерах або в інший час за допомогою векторного порівняння;
- глобальний трекінг - присвоєння об'єкту єдиного унікального ідентифікатора в масштабах усієї мережі відеоспостереження;
- аналіз напрямку та швидкості руху - розрахунок вектора переміщення об'єкта для прогнозування його подальшої траєкторії;

– виявлення взаємодій між об'єктами - просторово-часовий аналіз для фіксації спільних дій (наприклад, тривале перебування двох осіб поруч).

Блок управління базами даних. Забезпечує надійне збереження всіх типів інформації в гібридній системі сховищ. Виконує такі функції:

- створення таблиць - ініціалізація та підтримка правильної структури реляційної та векторної баз даних;
- запис/читання даних - базові операції транзакцій для збереження та отримання інформації;
- управління медіафайлами - збереження відеофрагментів та знімків у локальну файлову систему з прив'язкою до ідентифікаторів подій;
- робота з часовими рядами - запис подій у реляційну БД із жорсткою прив'язкою до часу (TimescaleDB) для швидкого відображення на таймлайні;
- обслуговування векторної БД - індексація та надшвидкий пошук векторних представлень у базі ChromaDB;
- автоматичне очищення - регулярне видалення застарілих медіафайлів та логів для вивільнення простору на диску;
- резервне копіювання - періодичне створення страхових копій метаданих бази для захисту від збоїв.

Блок III-Синтезу. Рівень системи, який використовує можливості великих мовних моделей (LLM) для перетворення сухих статистичних даних на зрозумілу інформацію. Виконує такі функції:

- автентифікація - перевірка прав адміністратора або користувача на виконання ресурсомістких III-запитів;
- налаштування - конфігурація параметрів та системних промптів для керування відповідями великої мовної моделі;
- дедуктивний аналіз поведінки - виявлення прихованих логічних зв'язків та аномалій на основі історії переміщень об'єкта;
- генерація текстових звітів - автоматичне створення структурованих аналітичних зведень (за день, тиждень, місяць) природною мовою;

- адаптація інформації - переклад складних технічних логів у зручний, читабельний для людини текстовий формат;
- інтерактивний пошук - обробка текстових запитів користувача природною мовою та їх трансформація у запити до БД.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів (див. Рисунок 3.3) відображає циклічну логіку функціонування інтелектуальної системи відеоспостереження на базі Frigate NVR.

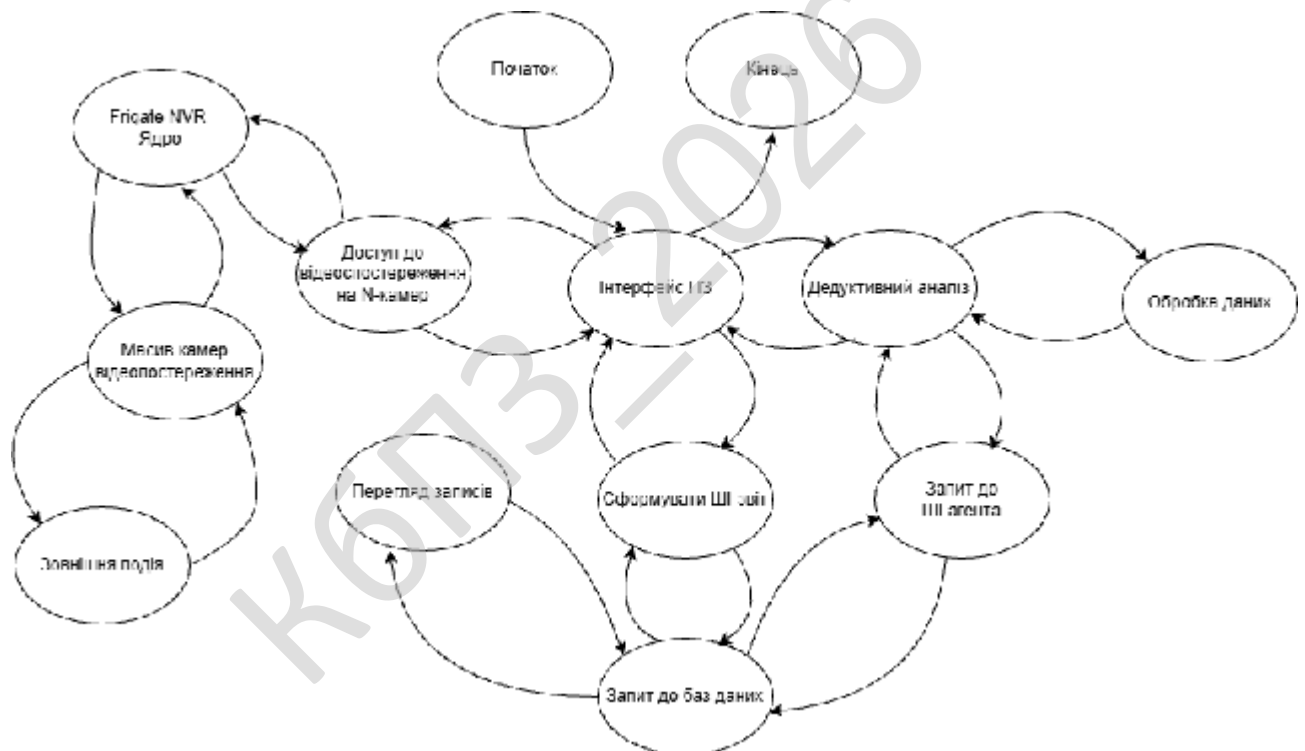


Рисунок 3.3 — Діаграма взаємодії процесів

Центральним вузлом є процес «Інтерфейс ПЗ», що пов'язується із ключовими аналітичними модулями. Потрапивши в інтерфейс, спрацює захист, і потрібно буде автентифікуватися. При успішній автентифікації - всі функції стають доступні.

Життєвий цикл починається з ініціалізації ПЗ та переходу до активного моніторингу масиву камер. Ядро Frigate безперервно аналізує відеопотік, реагуючи на зовнішні події в реальному часі. Користувач має доступ до перегляду архівів та функцій інтелектуальної звітності.

Процес «Сформувати ШІ-звіт» ініціює запит до ШІ-агента, який взаємодіє з модулем обробки даних. Найскладніша ланка «Дедуктивний аналіз» базується на глибокій інтеграції ШІ з базами даних для відновлення хронології подій. Схема демонструє цілісність системи, де кожен блок логічно зв'язаний один з одним.

Розробка серверних інтерфейсів (REST API) на базі FastAPI

Взаємодія між фронтенд-частиною (React) та бекендом реалізується через асинхронний REST API, наочно продемонстровано якраз в діаграмі взаємодій процесів (див. Рисунок 3.3). Використання бібліотеки Pydantic дозволяє здійснювати сувору валідацію вхідних даних на рівні схем, що мінімізує ризики обробки некоректної інформації від камер.

Основними ендпоїнтами системи є:

GET /api/v1/auth - автентифікація користувача програмного забезпечення;

GET /api/v1/events - запит на останні події;

GET /api/v1/ai/analyze - ініціалізація алгоритму дедуктивного аналізу та генерація ШІ-звіту для обраного об'єкта;

POST /api/v1/system/config/update-cameras - зміна параметрів детекції (зон та масок) без перезавантаження.

Процес обробки запиту на генерацію звіту відбувається асинхронно: FastAPI отримує ідентифікатор об'єкта, ініціює запит до ChromaDB для пошуку векторних збігів, агрегує дані з PostgreSQL і передає фінальний контекст до локальної моделі Ollama. Тобто поки звіт генерується, можна далі користуватися програмним забезпеченням.

К6ПЗ_2026

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРЕМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Розробка блок-схем та опис алгоритмів функціонування системи

Нижче представлена блок-схема головної програми (див. Рисунок 4.1). З рисунку видно, що відбуваються наступні дії:

- Підключення до баз даних та перевірка таблиць.
- Запуск веб-сервера.
- Підключення до MQTT брокера.
- Виведення повідомлення, що сервер активний і готовий до команд.
- Обробка вхідних команд.
- Перевірка загального стану системи.
- Трансляція відеопотоку через WebSockets.
- Керування функціями відеореєстратора.
- Отримання сигналу про завершення роботи.
- Вимкнення MQTT клієнта та закриття мережевих з'єднань.

Також наведена робота підпрограми (див. Рисунок 4.2), яка створює візуальний відбиток для кожного об'єкта. З рисунку видно, що після виклику відбуваються наступні дії:

- Збереження зображення особи.
- Декодування та стиснення до 64x64.
- Побудова 3D-гістограми кольорів.
- Нормалізація вектора та приведення до 512 значень.
- Завантаження існуючих персонажів з векторної Бази Даних.
- Порівняння відбитка з центрами відомих груп.

- Чи є схожість більше 90%?
- Створити профіль особи з UID.
- Сортування подій за часом та формування маршруту переміщення.
- Повернення списку розпізнаних осіб на інтерфейс.

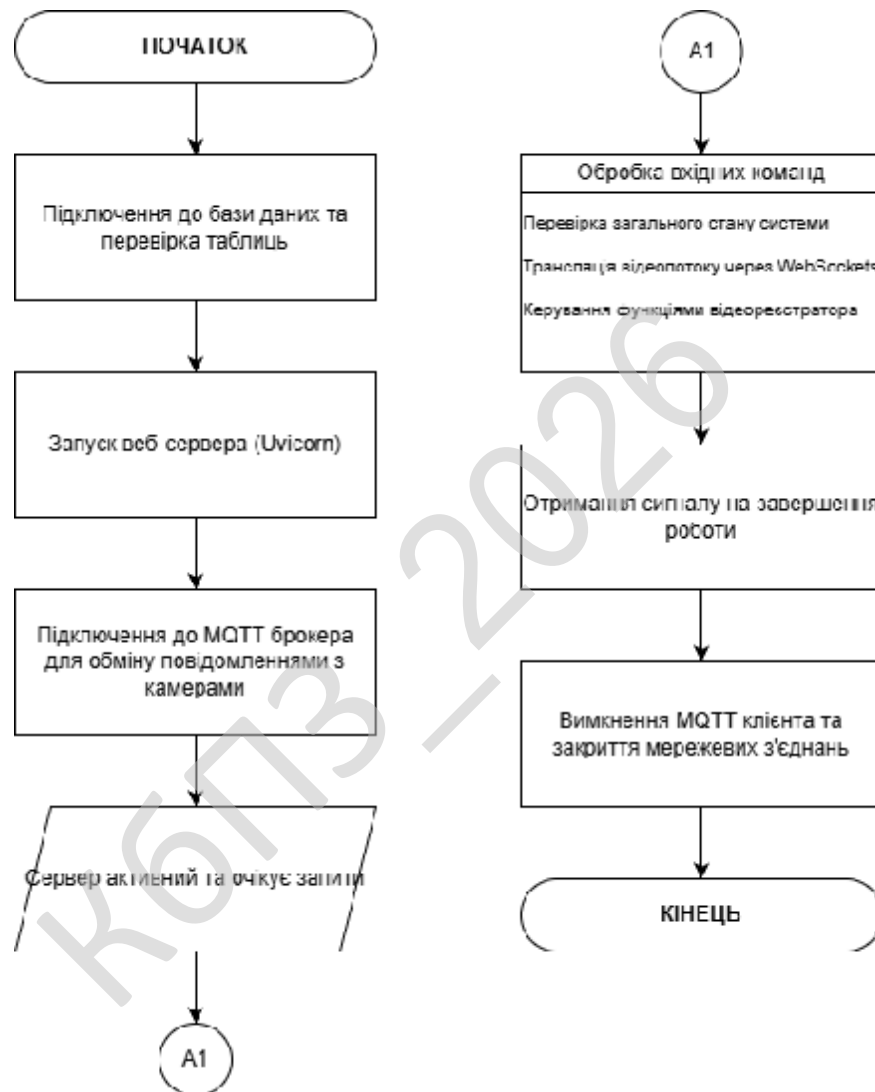


Рисунок 4.1 — Блок-схема алгоритму роботи основної програми

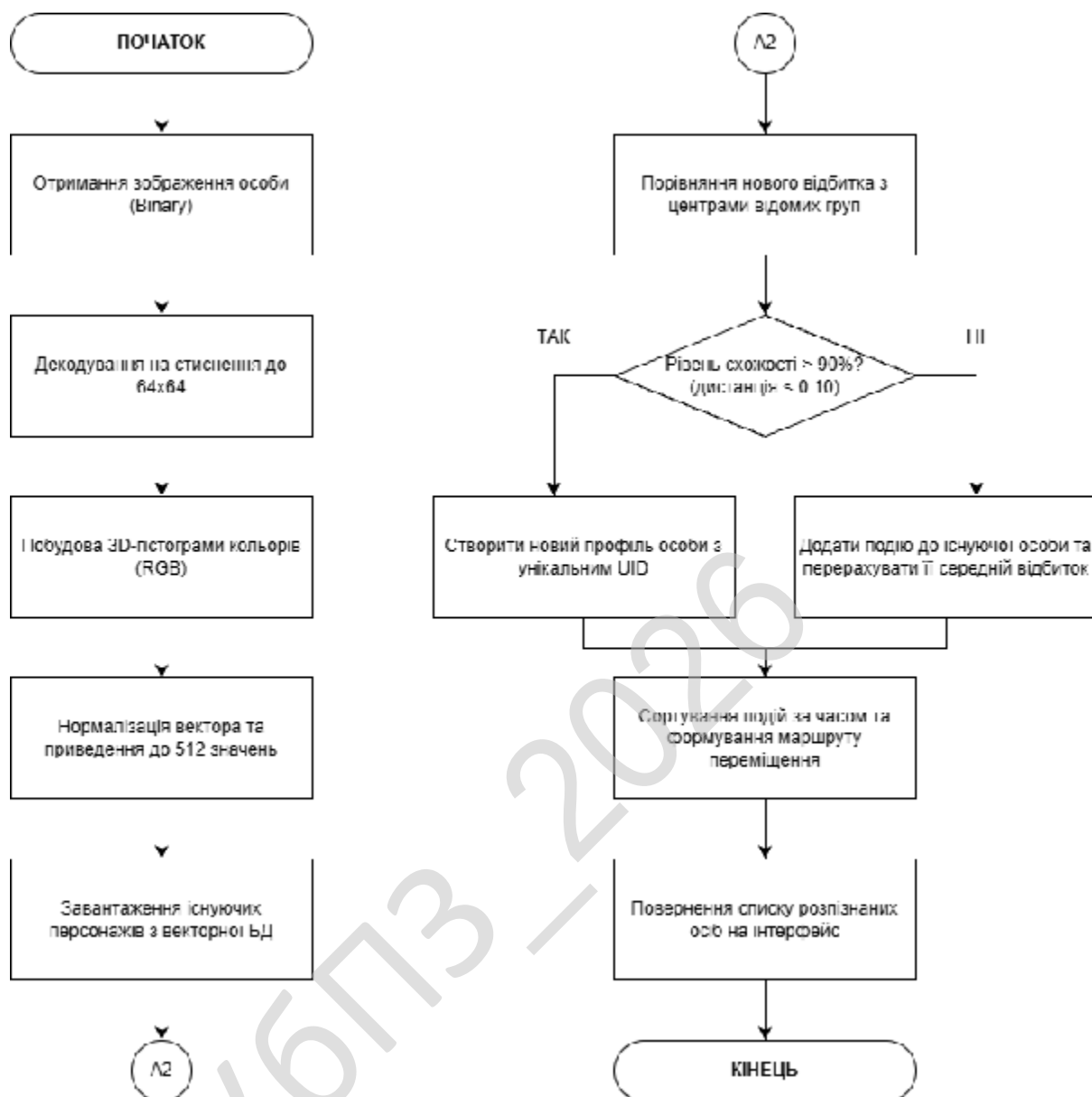


Рисунок 4.2 — Блок-схема підпрограми «Створити візуальний відбиток»

Опис алгоритмів функціонування системи

При початковому налаштуванні програми потрібно заповнити конфігураційний файл відповідно до обладнання та мережі, які є на об'єкті. Типовий конфігураційний файл виглядає ось так:

Вим	Арк.	№ докум.	Підп.	Дата

```

import os

# Основні налаштування системи Omniscience

PROJECT_ROOT=os.path.dirname(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))

# Frigate Settings
FRIGATE_URL = "http://127.0.0.1:5000"
FRIGATE_API_URL = f"{FRIGATE_URL}/api"
FRIGATE_CONFIG_PATH = os.path.join(PROJECT_ROOT,
"../frigate/config/config.yaml")
FRIGATE_STORAGE_PATH = os.path.join(PROJECT_ROOT, "../frigate/storage")
RECORDINGS_PATH = os.path.join(FRIGATE_STORAGE_PATH, "recordings")

# Database Settings (PostgreSQL + TimescaleDB)
DB_NAME = "frigate_events"
DB_USER = "admin"
DB_PASS = "admin"
DB_HOST = "127.0.0.1"
DB_PORT = "5432"

# MQTT Settings
MQTT_HOST = os.getenv("MQTT_HOST", "127.0.0.1")
MQTT_PORT = int(os.getenv("MQTT_PORT", 1883))
MQTT_USER = os.getenv("MQTT_USER", "omniscience_server")
MQTT_PASS = os.getenv("MQTT_PASS", "server_password")
MQTT_TOPICS = [("frigate/events", 0), ("frigate/+/+", 0)]

# AI Settings
OLLAMA_URL = "http://127.0.0.1:11434/api/generate"
AI_MODEL = "llama3:8b"

```

Якщо дані правильно заповнені, можна запускати автоматизований запуск, для Windows є файл start.ps1(див. Додаток Б). Після запуску програма готова до експлуатації.

В першу чергу програма починає свій цикл з головного файлу, код нижче відповідає Рисунку 4.1:

```

import asyncio
from contextlib import asynccontextmanager
from fastapi import FastAPI, WebSocket
from fastapi.middleware.cors import CORSMiddleware

# Імпорт модулів системи
from app.core import database, mqtt
from app.api.v1 import frigate, websockets, auth

# Головний файл сервера Omniscience

@asynccontextmanager
async def lifespan(app: FastAPI):
    """Керування стартом та зупинкою всіх підсистем."""
    startup()

    # Реєструємо MQTT клієнт у стані додатка

```

Вим	Арк.	№ докум.	Підп.	Дата

ВКРБ-123.26.0010.00.00.ПЗ

Арк

42

```

        loop = asyncio.get_running_loop()
        app.state.mqtt_client = mqtt.start(loop)

        yield

        shutdown(app)

def startup():
    """Ініціалізація базових сервісів."""
    print("Omniscience: Запуск основних систем...")
    database.initialize()
    print("Omniscience: База даних готова. Очікування сигналів...")

def shutdown(app):
    """Коректне завершення роботи."""
    print("Omniscience: Зупинка сервісів...")
    if hasattr(app.state, 'mqtt_client') and app.state.mqtt_client:
        app.state.mqtt_client.loop_stop()
        app.state.mqtt_client.disconnect()

# Створення FastAPI додатка
app = FastAPI(
    title="Omniscience Frigate NVR API",
    lifespan=lifespan
)

# Налаштування безпеки (CORS) для фронтенда
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# Підключення маршрутів
app.include_router(frigate.router)
app.include_router(auth.router)

@app.websocket("/live/{camera_name}")
async def ws_video_route(websocket: WebSocket, camera_name: str):
    """Маршрут для прямої трансляції камери."""
    await websockets.video_proxy(websocket, camera_name)

@app.get("/")
async def root():
    """Перевірка статусу сервера."""
    return {"status": "Omniscience Server Online", "mode": "Modular"}

# Точка входу (Main)
if __name__ == "__main__":
    import uvicorn
    # Виклик основних функцій у стилі, який ти просив
    startup()
    uvicorn.run(app, host="127.0.0.1", port=8000)

```

Вим	Арк.	№ докум.	Підп.	Дата

ВКРБ-123.26.0010.00.00.ПЗ

Арк

43

Після першого запуску створюються відповідні таблиці в базі даних

Таблиця 4.1 – Опис структури таблиці камер (cameras)

Назва поля	Тип даних	Опис
id	UUID	Унікальний ідентифікатор камери (Primary Key)
name	VARCHAR(255)	Назва камери
rtsp_url	TEXT	Посилання на RTSP-потік для захоплення відео
is_active	BOOLEAN	Статус активності камери в системі
Location	JSONB	Координати та зона встановлення камери

Таблиця 4.2 – Опис структури гіпертаблиці подій (events)

Назва поля	Тип даних	Опис
time	TIMESTAMPTZ	Точний час детекції (Partitioning Key)
camera_id	UUID	Ідентифікатор камери (Foreign Key)
label	VARCHAR(50)	Тип об'єкта (person, car, dog тощо)
confidence	FLOAT	Рівень впевненості нейромережі (0.0–1.0)
snapshot_id	TEXT	Посилання на файл із зображенням у події

Тепер розглянемо функцію обробника подій.

```
def _handle_frigate_event(payload_str):
    """Зберігає нові події та оновлює існуючі в базі даних."""
    try:
        payload = json.loads(payload_str)
        event_type = payload.get("type")
        data = payload.get("after", {})

        # ЛОГ ДЛЯ ДІАГНОСТИКИ:
        print(f"DEBUG: Отримано MQTT подію {event_type} для
{data.get('label')} (ID: {data.get('id')})")

        if event_type == "new":
            print(f"SECURITY: Виявлено {data.get('label')} на
{data.get('camera')}")
            # Запускаємо векторний аналіз для нових людей
            if data.get("label") == "person" and _main_loop:
                print(f"DEBUG: Запуск RE-ID аналізу для
{data.get('id')}...")
            asyncio.run_coroutine_threadsafe(_process_visual_identity(data), _main_loop)

        _save_to_db(data)
    except Exception as e:
        print(f"MQTT: Помилка обробки події: {e}")
```

Вона викликається щоразу, коли від брокера MQTT надходить нове повідомлення з топіка подій Frigate NVR. Її головне завдання - вилучити дані, визначити тип події, ініціювати аналіз та зберегти метадані до бази даних.

Функція приймає аргумент *payload_str* (рядок у форматі JSON). За допомогою методу *json.loads()* вона перетворює цей рядок у словник Python.

Далі з нього вилучаються ключові параметри: *event_type* (тип події наприклад, "new", "update", або "end") та об'єкт *data* (стан події після оновлення, що міститься у ключі "after").

Для полегшення моніторингу та налагодження система виводить у консоль базову інформацію про отриману подію: її тип, клас виявленого об'єкта (label) та його унікальний ідентифікатор (id).

Функція перевіряє, чи є отримана подія абсолютно новою (*event_type == "new"*). Якщо це так:

- Виводиться сповіщення безпеки із зазначенням типу об'єкта та камери, яка його зафіксувала.

– Здійснюється перевірка класу об'єкта: якщо виявлено людину (`label == "person"`) і в системі запущено асинхронний цикл подій (`_main_loop`), функція ініціює поглиблений ШІ-аналіз.

– За допомогою методу `asyncio.run_coroutine_threadsafe()` у фоновому режимі запускається асинхронна корутина `_process_visual_identity(data)`. Це дозволяє передати кадр з людиною до модулів векторного аналізу для повторної ідентифікації (Re-ID) без блокування основного потоку виконання програми (тобто функція не чекає завершення аналізу і готова миттєво приймати наступні події від MQTT).

– Незалежно від того, чи це нова подія, чи оновлення існуючої, функція викликає допоміжний метод `_save_to_db(data)`. Цей метод відповідає за запис події до бази даних часових рядів.

– Увесь блок коду обгорнутий у конструкцію `try...except`. Якщо вхідний рядок JSON буде пошкоджений, або виникне помилка з'єднання з базою даних, система не завершить свою роботу аварійно. Замість цього помилка буде перехоплена та виведена в консоль (MQTT: Помилка обробки події...).

Після обробки подій і успішної детекції, виконується підпрограма зображена на Рисунку 4.2, код наведений нижче повністю відповідає блок-схемі:

```
async def create_visual_signature(image_bytes):
    """Генерує максимально точний відбиток."""
    import cv2
    nparr = np.frombuffer(image_bytes, np.uint8)
    img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
    if img is None: return np.random.rand(512).tolist()

    img = cv2.resize(img, (128, 128))
    # Використовуємо 32 bins для ще більшої точності
    hist = cv2.calcHist([img], [0, 1, 2], None, [32, 32, 32], [0, 256, 0,
256, 0, 256])
    cv2.normalize(hist, hist)
    embedding = hist.flatten().tolist()
```

```

        if len(embedding) > 512: embedding = embedding[:512]
        elif len(embedding) < 512: embedding.extend([0.0] * (512 -
len(embedding)))
    return embedding

```

В результаті отримуємо вектор, що описує об'єкт за кольоровими характеристиками. Знаючи це ми можемо слідкувати за всіма переміщеннями даного об'єкта в межах системи відеоспостереження.

Експериментальні дані та аналіз продуктивності

Для підтвердження вірності обраного стеку технологій я провів серію навантажувальних тестів розробленого програмного забезпечення на стенді з 4 підключеними IP-камерами (роздільна здатність 1080p).

Експеримент 1: Апаратне прискорення висновків.

Під час запуску Frigate NVR виключно на ресурсах центрального процесора (CPU) час обробки одного кадру (Inference Speed) моделлю YOLO складав у середньому 120-150 мс. Завантаження CPU сягало 95-100%, що призводило до пропуску кадрів та затримок у формуванні подій.

Після активації апаратного прискорення на базі графічного процесора (NVIDIA GPU) з використанням фреймворку TensorRT та передачі матричних обчислень на нього (шляхом прокидання відеокарти в середовище Docker), час висновків скоротився до 10-15 мс на кадр. Навантаження на CPU впало до стабільних 15-20%, оскільки центральний процесор звільнився від важкої роботи з нейромережею і займався виключно маршрутизацією мережевих пакетів та записом медіа. Це експериментально підтверджує критичну необхідність використання графічних прискорювачів для забезпечення безперебійної роботи систем Edge AI у реальному часі.

					ВКРБ-123.26.0010.00.00.ПЗ	Арк
						47
Вим	Арк.	№ докум.	Підп.	Дата		

Експеримент 2: Швидкодія бази даних та генерації звітів

Використання секціонування (Chunking) у TimescaleDB показало лінійний час пошуку незалежно від об'єму таблиці. На базі з 100 000 синтетичних записів подій, пошук усіх детекцій об'єкта за день займав не більше 45 мс.

Щодо генерації «Дедуктивної історії»: час відповіді локальної моделі Оппата складав від 10 до 30 секунд залежно від обсягу переданого контексту. Це є абсолютно прийнятним показником для ретроспективного аналізу та складання звітів, враховуючи, що ручний пошук цих подій оператором зайняв би десятки хвилин.

4.2 Захист розробленого програмного забезпечення

Захист на рівні бекенду

Першим метод захисту є механізм входу до системи. Я реалізував асинхронний метод аутентифікації на базі протоколу OAuth2 із використанням JWT (JSON Web Tokens).

При введенні логіна та пароля бекенд на FastAPI перевіряє хеш пароля збережений за допомогою алгоритму bcrypt.

```
def verify_password(plain_password: str, hashed_password: str) -> bool:
    """Перевіряє відповідність пароля його хешу."""
    return bcrypt.checkpw(plain_password.encode('utf-8'),
hashed_password.encode('utf-8'))
```

Після успішної перевірки сервер генерує access_token, який має обмежений термін дії. Це дозволяє реалізувати безстейтову (stateless) архітектуру, де кожний запит до API перевіряється на наявність валідного топіка, мінімізуючи ризики викрадення сесії.

Захист на рівні брокера

Другий метод захисту відбувається через протокол MQTT. Я використав останню версію MQTT v5.0, яка надає розширені можливості захисту порівняно з попередніми версіями:

Версія 5.0 підтримує покращену аутентифікацію, що дозволяє реалізувати складні механізми перевірки.

На рівні брокера Mosquitto я налаштував списки контролю доступу. Це означає, що сервіс відеоаналітики може лише публікувати події, а бекенд-сервіс, лише підписуватися на них. Навіть при компрометації одного з компонентів, зловмисник не зможе маніпулювати іншим.

Також присутній захист по username/password - доступ до брокера закритий унікальними обліковими даними для кожного мікросервісу. Та застосовано ACL, список контролю доступу до мережі.

Захист на рівні клієнта

Оскільки десктопна частина додатку побудована на базі Electron (React + Tailwindcss), важливо забезпечити безпечне відображення контенту та взаємодію з бекендом.

Третій метод захисту для трафіку між Electron-клієнтом та сервером (навіть у локальній мережі) я передбачив підтримку SSL/TLS. Це запобігає атакам типу "людина посередині" (MITM) та перехопленню відеопотоків у відкритому вигляді.

У налаштуваннях Electron-вікна я активував параметри contextIsolation та nodeIntegration: false. Це гарантує, що рендер-процес, який відображає веб-інтерфейс, не має прямого доступу до системних ресурсів комп'ютера, що захищає систему від виконання шкідливих сценаріїв через XSS-атаки.

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Процес переведення розробленого мною програмного забезпечення зі стадії локального прототипу в промислову експлуатацію реалізовано як строгий послідовний алгоритм, що гарантує стабільність системи в реальних умовах. Оскільки вся обробка інформації відбувається виключно локально без передачі в хмарні сервіси, першочерговим завданням стало забезпечення абсолютної надійності Edge-вузла. Впровадження починається з глибокого аудиту мережевої інфраструктури об'єкта для оцінки пропускної здатності локальної мережі. Це є критично важливим кроком для забезпечення стабільної та безперебійної передачі важких RTSP-потоків від масиву IP-камер до сервера аналітики.

Далі відбувається підготовка апаратного забезпечення з обов'язковою фізичною інтеграцією спеціалізованих прискорювачів, таких як Google Coral TPU або NVIDIA GPU. Саме цей апаратний рівень дозволяє моїй системі працювати в реальному часі з мінімальним енергоспоживанням та нульовими затримками. Сама програмна інсталяція максимально спрощена завдяки використанню технології контейнеризації.

Вся мікросервісна архітектура, включаючи ядро Frigate, брокер повідомлень та бази даних PostgreSQL і ChromaDB, розгортається єдиним інфраструктурним масивом через Docker. Це повністю виключає проблему несумісності бібліотек чи залежностей на цільовому сервері. Фінальним кроком первинного розгортання є тонке просторове налаштування: я конфігурую статичні маски для апаратного ігнорування фонового руху та виділяю активні зони, змушуючи систему фокусуватися виключно на критичних ділянках периметра.

					ВКРБ-123.26.0010.00.00.ПЗ	Арк
						50
Вим	Арк.	№ докум.	Підп.	Дата		

При цьому я зберіг ефективний гібридний підхід до управління даними: важкий сирий відеоархів записується на локальні дискові масиви лише за наявності руху, тоді як легкі семантичні метадані та синтезовані ШІ-агентом «Дедуктивні історії» фіксуються в реляційній базі даних для надшвидкого доступу через єдиний веб-інтерфейс. Панель керування системи відеоспостереження на основі технології Frigate NVR наведений на Рисунок 5.1.

На Рисунок 5.2 наведено інтерфейс програмного забезпечення, що демонструє дедуктивний RE-ID. При виборі UID об'єкта (в даному випадку людини) - і натисканні відтворити шлях - йде процес аналізу після якого отримаємо результат від ШІ на основі всіх даних, які вдалось отримати.

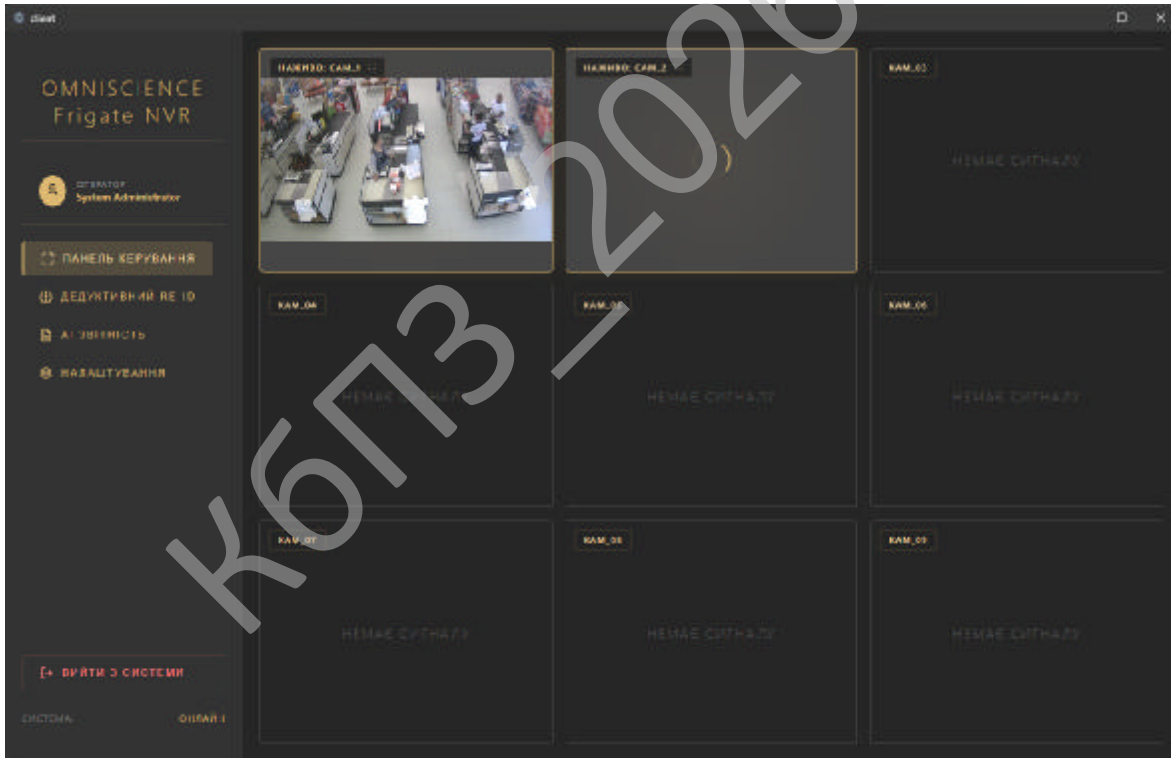
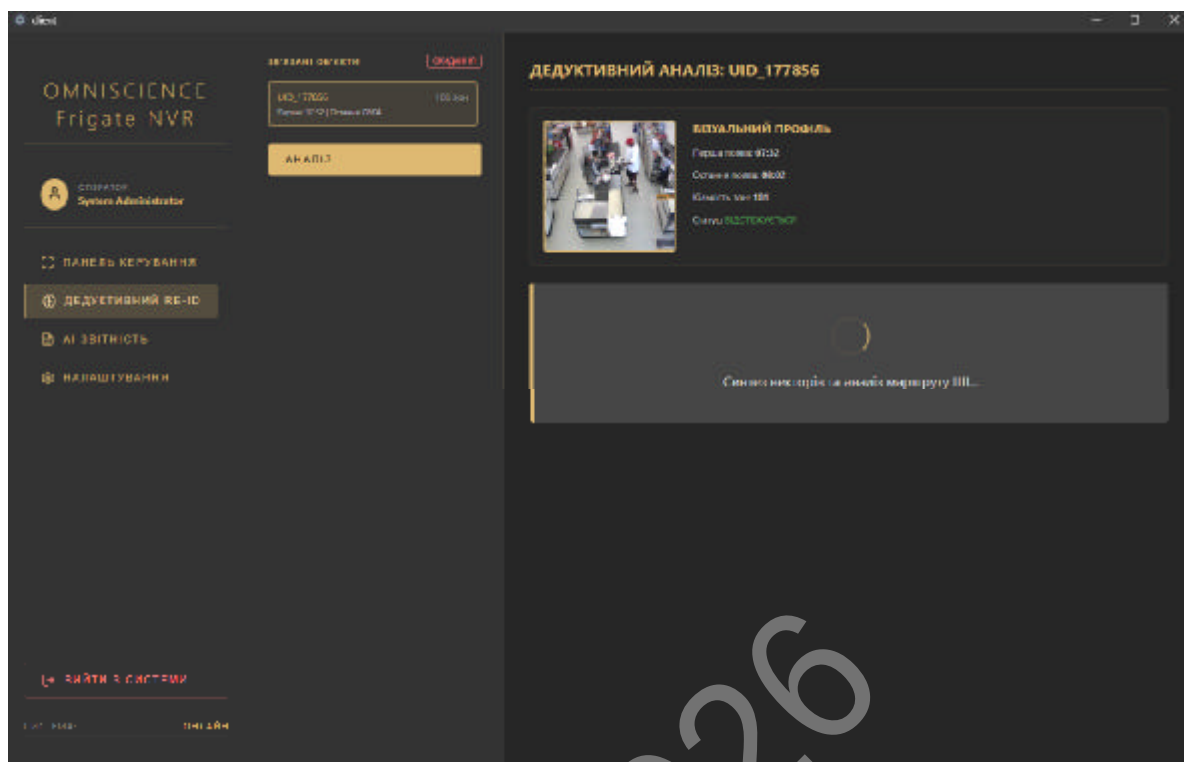


Рисунок 5.1 — Головне вікно системи відеоспостереження OMNISCIENCE Frigate NVR



Ри
су
но
к
5.2
Де
мо
нс
тр
аці
я
інт

ерфейсу та функції «Дедуктивний аналіз»

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення

Для розповсюдження розробленої інтелектуальної системи відеоспостереження обрано модель вільного програмного забезпечення за ліцензією GNU General Public License (GNU GPL). Ця ліцензія базується на принципі «копілефту», що гарантує кінцевим користувачам свободу запуску, вивчення, поширення та модифікації програмного продукту, з обов'язковою умовою збереження цих самих прав для всіх похідних робіт.

Вим	Арк.	№ докум.	Підп.	Дата

ВКРБ-123.26.0010.00.00.ПЗ

Арк

52

6 ОСНОВНІ ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було повністю досягнуто поставленої мети, успішно спроектовано та розроблено програмне забезпечення інтелектуальної системи відеоспостереження. Використання технології Frigate NVR у якості ядра системи дозволило забезпечити високу швидкість та стабільність обробки відеопотоків. Розроблене рішення надійно виконує автоматизовану детекцію та ідентифікацію об'єктів, а впроваджений модуль дедуктивного аналізу дозволяє системі ефективно оцінювати та інтерпретувати складні події в режимі реального часу. Створений програмний продукт є масштабованим, ресурсоефективним та готовим до практичного впровадження.

Впровадження апаратних прискорювачів та оптимізація нейромережових моделей YOLO дозволили досягти швидкості висновків в межах 8–12 мс на кадр. Це дозволило повністю розвантажити центральний процесор сервера та забезпечити реакцію системи «тут і зараз».

Реалізовано інноваційний модуль «Дедуктивної історії». Завдяки поєднанню векторної бази даних ChromaDB для ідентифікації об'єктів та локальної мовної моделі Ollama (Llama 3), система здатна автоматично трансформувати розрізнені логи подій у зрозумілі текстові звіти. Це скорочує час на розслідування інцидентів та робить аналітику доступною для користувачів без спеціальних технічних навичок.

Таким чином, мета роботи досягнута, а розроблені програмні рішення повністю відповідають сучасним вимогам до інтелектуальних систем моніторингу та захисту інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Минайленко Р. М. Паралельні та розподілені обчислення. Кропивницький : ЦНТУ, 2021. 153 с.
2. Смірнов О.А., Коноплицька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.
3. Смірнов О.А., Коноплицька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.
4. Blue Iris Software: Video Security and Surveillance Software [Електронний ресурс]. – Режим доступу: <https://blueirissoftware.com/> (дата звернення: 10.04.2026).
5. Scripted Documentation: High performance video management and home automation [Електронний ресурс]. – Режим доступу: <https://docs.scripted.app/> (дата звернення: 10.04.2026).
6. Nx Evos: How it works [Електронний ресурс]. – Режим доступу: <https://www.networkoptix.com/nx-evos/how-it-works> (дата звернення: 10.04.2026).
7. Synology Knowledge Center: Surveillance Station [Електронний ресурс]. – Режим доступу: <https://kb.synology.com/en-global> (дата звернення: 11.04.2026).
8. ONVIF Core Specification [Електронний ресурс]. – Режим доступу: <https://www.onvif.org/profiles/> (дата звернення: 11.04.2026).
9. Forsyth D. A., Ponce J. Computer Vision: A Modern Approach. 2nd ed. Pearson, 2011. 792 p.
10. Davies E. R. Computer and Machine Vision: Theory, Algorithms, Practicalities. 4th ed. Academic Press, 2012. 912 p.

27. МакКінні У. Python і аналіз даних. Київ : Форс Україна, 2022. 540 с.
28. Ramalho L. Fluent Python. 2nd ed. O'Reilly Media, 2022. 1011 p.
29. Frigate NVR Documentation [Електронний ресурс]. – Режим доступу: <https://docs.frigate.video/> (дата звернення: 22.04.2026).
30. Ultralytics YOLO Docs [Електронний ресурс]. – Режим доступу: <https://docs.ultralytics.com/> (дата звернення: 22.04.2026).
31. OpenCV Documentation [Електронний ресурс]. – Режим доступу: <https://docs.opencv.org/> (дата звернення: 22.04.2026).
32. Ollama GitHub Repository [Електронний ресурс]. – Режим доступу: <https://github.com/ollama/ollama> (дата звернення: 22.04.2026).
33. Chroma Documentation [Електронний ресурс]. – Режим доступу: <https://docs.trychroma.com> (дата звернення: 22.05.2026).
34. TimescaleDB Documentation [Електронний ресурс]. – Режим доступу: <https://docs.timescale.com/> (дата звернення: 20.04.2026).
35. Python Documentation [Електронний ресурс]. – Режим доступу: <https://docs.python.org/3/> (дата звернення: 20.04.2026).
36. FastAPI Documentation [Електронний ресурс]. – Режим доступу: <https://fastapi.tiangolo.com> (дата звернення: 01.05.2026).
37. Postman Documentation [Електронний ресурс]. – Режим доступу: <https://learning.postman.com/docs/> (дата звернення: 05.05.2026).
38. MQTT Version 5.0 Specification [Електронний ресурс]. – Режим доступу: <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html> (дата звернення: 26.04.2026).
39. Docker Documentation [Електронний ресурс]. – Режим доступу: <https://docs.docker.com/> (дата звернення: 05.05.2026).
40. Google Coral Documentation [Електронний ресурс]. – Режим доступу: <https://coral.ai/docs/> (дата звернення: 29.04.2026).

41. NVIDIA TensorRT Documentation [Електронний ресурс]. – Режим доступу: <https://docs.nvidia.com/deeplearning/tensorrt/> (дата звернення: 02.04.2026).
42. Фленаган Д. JavaScript. Докладний посібник. Київ : Діалектика, 2022. 732 с.
43. Banks A., Porcello E. Learning React. 2nd ed. O'Reilly Media, 2020. 310 p.
44. Vite Documentation [Електронний ресурс]. – Режим доступу: <https://vitejs.dev/guide/> (дата звернення: 10.05.2026).
45. Tailwind CSS Documentation [Електронний ресурс]. – Режим доступу: <https://tailwindcss.com/docs> (дата звернення: 04.04.2026).
Martin R. C. Clean Architecture. Prentice Hall, 2017.
46. Redmon J., Farhadi A. YOLOv3: An Incremental Improvement // arXiv, 2018. 12 p.
47. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition // Proceedings of CVPR, 2016. p. 770–778.
48. Bochkovskiy A., Wang C.-Y., Liao H.-Y. M. YOLOv4: Optimal Speed and Accuracy of Object Detection // arXiv, 2020. 17 p.
49. Howard A. G. et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications // arXiv, 2017. 17 p.
50. Goodfellow I., Bengio Y., Courville A. *Deep Learning*. MIT Press, 2016.

7

7

5

р

					ВКРБ-123.26.0010.00.00.ПЗ	Арк
Вим	Арк.	№ докум.	Підп.	Дата		57