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
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи відеоспостереження за технологією VSaaS”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Дар‘єв Б.В.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук
_____ Лисенко І.А.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *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
« 17 » січня 2025 року

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

Дар‘єву Богдану Вікторовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи відеоспостереження за технологією VSaaS*

2. Керівник роботи *Лисенко Ірина Анатоліївна, канд. техн. наук*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 46-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи відеоспостереження за технологією VSaaS*

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 « 17 » січня 2025 р.

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	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Лисенко І.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Дар‘єв Б.В.
(прізвище та ініціали)

АНОТАЦІЯ

Дар‘єв Б.В. Програмне забезпечення системи відеоспостереження за технологією VSaaS. 123 Комп’ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи відеоспостереження за технологією VSaaS.

Метою розробки є програмне забезпечення системи відеоспостереження за технологією VSaaS.

Результат роботи – програмна реалізація системи відеоспостереження за технологією VSaaS.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп’ютерна інженерія, відеоспостереження, VSaaS

ABSTRACT

Dar'iev B.V. Software for a video surveillance system using VSaaS technology. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for a video surveillance system using VSaaS technology.

The purpose of the development is software for a video surveillance system using VSaaS technology.

The result of the work is a software implementation of a video surveillance system using VSaaS technology.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software are provided.

The program can be used on a PC with OS Windows 10/11.

The program was developed in the Python environment.

Keywords: computer engineering, video surveillance, VSaaS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

DS – DirectShow мультимедійне розширення Microsoft;

TrFl – TranformFilter – Трансформ-фільтр;

RnFl – RendererFilter – рендер-фільтр;

SourceFilter – SourceFilter – фільтр джерела;

IC – Intelligent Connect – інтелектуальне з'єднання;

ООП – об'єктно-орієнтоване програмування;

DSP – бібліотека компонентів для розробки мульдемедійних програм;

VWINDOW – вертикальна синхронізація;

HWINDOW – горизонтальна синхронізація;

PLL – цикл Блокування Стадії;

USB – Universal Serial Bus – універсальна послідовна шина;

URL – universal resource locator – локатор ресурсів інтернет;

HTML – мова розмітки гіпертекстових документів;

ОС – операційна система;

ПЕОМ – персональна електронно обчислювальна машина;

КС – комп’ютерна система;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

ВСТУП

Актуальність теми. Відеоспостереження як послуга (VSaaS) – це хмарне рішення для відеоспостереження, яке дозволяє операторам служби безпеки віддалено контролювати кілька приміщень за допомогою камер, які надсилають відео в центральне хмарне сховище через Інтернет. Звідти відео можна отримати віддалено з будь-якого пристрою, підключеного до Інтернету. Це дозволяє операторам служби безпеки стежити за своїми приміщеннями, де б вони не були, без необхідності встановлювати фізичні системи спостереження на місці.

Найбільшою перевагою VSaaS є те, що він усуває потребу у дорогих локальних системах. Оскільки вам не потрібно встановлювати фізичну інфраструктуру чи наймати техніків для керування ними, ви заощаджуєте час і гроші, які в іншому випадку були б витрачені на підтримку цих систем.

Одним із недоліків VSaaS є те, що він вимагає підключення до Інтернету. Без надійного високошвидкісного доступу до Інтернету відеоматеріали не передадуться належним чином і можуть призвести до прогалин у безпеці.

Прив’язка до постачальника може бути ще однією серйозною проблемою для компаній, які покладаються на VSaaS, оскільки вони можуть опинитись із застарілою системою, якщо їхній постачальник припинить роботу або припинить пропонувати підтримку для свого продукту. Крім того, зміна постачальника може бути дорогим і трудомістким, оскільки під час переходу компаніям доводиться інвестувати в нове обладнання та програмне забезпечення.

Відеоспостереження як послуга (VSaaS) – це інноваційний та економічно ефективний спосіб віддаленого моніторингу приміщень. VSaaS пропонує масштабованість, гнучкість, розширені функції безпеки, такі як алгоритми шифрування та списки контролю доступу, щоб забезпечити безпеку даних у будь-який час, дозволяючи компаніям мати доступ у будь-який час. З іншого боку, існують деякі недоліки, пов’язані з VSaaS, такі як ризики блокування

постачальника, якщо зміна постачальника стає необхідною, або проблеми сумісності, які можуть виникнути під час оновлення апаратного чи програмного забезпечення.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи відеоспостереження за технологією VSaaS.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем відеоспостереження за технологією VSaaS.

– Дослідження системи відеоспостереження за технологією VSaaS.

– Програмна реалізація системи відеоспостереження за технологією VSaaS.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі відеоспостереження за технологією VSaaS.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи відеоспостереження за технологією VSaaS, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Платформа VSaaS була впроваджена в галузі безпеки, роздрібної торгівлі та транспорту. Наша технологія базується на існуючій інфраструктурі (HW і Network), додавши крайовий AI (крайовий, локальний або хмарний), ми можемо підключити наявні камери, контроль доступу, NVR/DVR, це дозволить нам контролювати мережу/пристрої, запис і зберігання, а також запускати відеоаналітику. Це чудовий вибір для систем камер і датчиків, які працюють у складних мережевих умовах (Wi-Fi і стільникові мережі або навіть низька пропускна здатність):

1. Підключення. Підключіть дані до камери за допомогою протоколів RTP/RTSP/HTTP/TCP.
2. Інтеграція IoT. Протоколи підключення IP і промислового обладнання на основі MQTT
3. Стійкість. Гібридне рішення, яке використовує потенціал використання межової та хмарної гібридної архітектури
4. Безпека. Зашифровані та безпечні процеси повідомлень між пристроями Edge, а також під час їх зв’язку з хмарною платформою. За допомогою панелі керування доступом регулюйте користувачів, отримайте доступ до даних і використовуйте їхні програми
5. Аналітика та ШІ. VSaaS спрощує реалізацію моделей нейронних мереж, здатних виявляти всі типи об’єктів і шаблонів на пристроях Edge за допомогою наших програм штучного інтелекту
6. Віддалене керування (FOTA). VSaaS дозволяє адміністраторам надсилати конфігурації, оновлення, виправлення тощо на пристрої Edge із хмарної веб-програми

1.2 Область застосування

Особливості VSaaS:

– Керуйте кількома камерами в простому середовищі, яке легко налаштувати.

– Сумісність із 99% існуючих інфраструктур.

– Обчислення Edge AI дозволяють знизити витрати на централізований центр обробки даних.

– Підключіться та переглядайте потокове відео на будь-якому пристрої.

– Може працювати паралельно з іншими системами VMS.

Технологія:

– Платформа не залежить від будь-якого хмарного постачальника (AWS, MS Azure, Google тощо).

– Платформа підключається до камер різних виробників (Hikvision, Dahua, Boch та ін.).

– Набір інструментів для створення додатків ШІ.

– Алгоритми AI, сумісні з кількома апаратними архітектурами.

– Масове та дистанційне керування додатками на пристроях Edge.

– API даних для розробників.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи відеоспостереження за технологією VSaaS, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Digifort

Інтелектуальне програмне забезпечення для відеоспостереження, яке використовує найкращі доступні технології та забезпечує найкращу продуктивність для задоволення будь-яких потреб. Повна платформа керування IP-відео, яка дає змогу керувати подіями та автоматизувати середовища. Він надійний, інтуїтивно зрозумілий і простий у налаштуванні та використанні. Повністю інтегрована автоматизована система зчитування номерних знаків із нашою системою керування відео. Автоматизоване рішення для зчитування номерних знаків із попередньо налаштованими діями та автоматизованими сповіщеннями. Віддалене керування, захоплення та запис екранів комп'ютерів під керуванням Windows. Ідеально підходить для кол-центрів, безголового керування сервером і керування програмним забезпеченням сторонніх розробників. Live Applied Intelligence без відеомоніторингу забезпечує проактивні дії для ситуацій запрограмованої поведінки з автоматичним створенням тривог і подій. Він також надає цінну статистичну інформацію, яку можна використовувати для програм бізнес-аналітики.

KiwiVision

Стає все важче зрозуміти повсякденність у складному світі. Операторам важко керувати інформацією через величезну кількість датчиків, камер, сайтів і пристроїв. Операторам важко відокремлювати та фільтрувати відповідні дані через велику кількість подій. KiwiVision™, відеоаналітика, допомагає вам зрозуміти, що відбувається навколо вас. Це автоматизує спостереження та

дозволяє покращити ваші операції. Це дозволить вашій групі безпеки приймати швидші та більш обґрунтовані рішення. Керування відео може зробити більше, ніж просто забезпечити спостереження. Відеоаналітика може допомогти вам перетворити дані з ваших камер у корисну інформацію, яка покращить вашу безпеку. Ви можете ефективно виконувати завдання з технічного обслуговування, швидше реагувати на критичні події та отримувати інформацію про бізнес, щоб покращити свою роботу. Ви також можете захистити конфіденційність і анонімність людей без шкоди для безпеки.

Athena

Athena надсилає сповіщення відповідним чиновникам, коли виявляється підвищена температура. Athena інтегрується з VMS, контролем доступу, розробниками програмного забезпечення для управління офісом, щоб централізувати всю відповідну інформацію. Athena співпрацює з такими надійними брендами, як Dell, Texas Instruments, Apple та іншими, щоб гарантувати, що у нас є найкраще апаратне забезпечення. Наші нейронні мережі були розроблені докторами наук у галузі комп'ютерного зору та глибокого навчання. Наше обладнання було розроблено з використанням високоякісних тепловізійних камер для виявлення коливань температури та пристрою HSRP (чорне тіло) для коливань навколишнього середовища. Athena доступна як окремий пристрій, корпоративна хмара або як комбінація з корпоративним локальним сервером. Корпоративне рішення Athena можна розгорнути відразу після прибуття. Це система plug-and-play, яка автоматично калібрує, конфігурує та налаштовує сама себе.

Xeoma

Модулі в Xeoma включають:

– розпізнавання номерних знаків, облич і емоцій, статі, віку, статі;

– розпізнавання типів об'єктів;

– виявлення зниклих або покинутих предметів;

– відсутність або наявність медичних масок або засобів безпеки;

– втручання в камеру або інші проблеми зі здоров'ям системи;

– виявлення натовпу, кольору та безладдя;

– підрахунок відвідувачів, пасажирів;

– відстеження PTZ, екскурсії;

– екран захоплення;

– теплова карта, плани поверхів;

– системи організації та доступу;

– інтеграція касових апаратів (POS);

– інтеграція систем розумного дому;

– та інше.

Безкоштовний ребрендинг, організація власного хмарного сервісу, необмежену кількість серверів, необмежену кількість серверів та інше. Маєте бізнес-потреби, пов’язані зі штучним інтелектом і/або аналізом звуку/відео? Ми можемо допомогти вам знайти правильне рішення! Ми також проводимо індивідуальну платну розробку для досягнення ваших бізнес-цілей. Xeoma пропонує безкоштовні ліцензії на купольні камери для тестування та має постійно безкоштовний режим із переглядом до 1000 камер та записом архівів із 4 камер!

Milestone XProtect

Milestone XProtect®, програмне забезпечення для керування відео (VMS), об’єднує всі компоненти відеоспостереження в ідеальне поєднання для створення рішення, яке захищає людей і майно сьогодні та завтра. Купуючи систему відеоспостереження, ви інвестуєте не тільки в апаратне та програмне забезпечення. Ви інвестуєте в безпеку своєї родини, будинку, бізнесу, активів і людей, щоб ви могли захистити найважливіше. Вибір XProtect для вашої основи означає, що ви обираєте VMS, розроблену інноваційними експертами з відеоспостереження, які працюватимуть над вирішенням ваших проблем сьогодні та в майбутньому. XProtect – це перевірене рішення для більш ніж 500 000 інсталяцій по всьому світу, від квіткових магазинів до стадіонів, університетів і міст. Він на ринку вже більше 20 років.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – високорівнева мова програмування, яку називають другою за популярністю в світі. Її використовують для розробки вебзастосунків, програмного забезпечення, машинного навчання. Python застосовують для вирішення робочих завдань у компаніях Google, Instagram, Facebook, IBM, NASA, Dropbox, Netflix та інших. Розробники цінують цю мову програмування за простоту у вивченні, ефективність та мультиплатформність.

Python – скриптова мова програмування з досить простим синтаксисом. Для розуміння достатньо порівняти принципи написання найпростішої програми, яка виводить на екран текстове повідомлення. Саме тому мова програмування Python більш доступна для новачків, а професіонали встигли адаптувати її для вирішення великої кількості завдань. Це мультиплатформне рішення, тому знання Python дає можливість працювати у різних сферах: від розробки мобільних застосунків до ігрової індустрії та штучного інтелекту.

У мови програмування динамічна типізація: є можливість передавати до функцій будь-який тип даних без попереднього вказання. Інтерпретованість дозволяє знаходити помилки у коді ще до повної збірки у робочий застосунок. При цьому Python дуже чітко дає зрозуміти, де та через що виникла помилка.

Це мова об'єктноорієнтованого програмування (ООП). Програмне забезпечення на Python оформлене у вигляді моделей, які можуть бути зібраними у пакети. Тип та структуру кожного об’єкта можна запитати під час виконання програми. Для кожного з об’єктів можна отримати всю інформацію щодо його внутрішньої структури. Окрім того:

- у мови логічний синтаксис, завдяки чому вихідний код легко читати та розуміти;
- гнучкість та масштабованість Python дозволяє адаптувати високорівневу логіку та розширяти складні застосунки, як тільки виникне така необхідність;

– розробка на Python у більшості випадків проходить швидше, ніж на інших мовах програмування;

– Python – інтерпретована мова програмування. Це значить, що код можна написати у будь-якому текстовому файлі на будь-якій платформі, і потім успішно запустити;

– у Python – колосальна спільнота однодумців. Тож будь-які складнощі конкретних розробників вирішуються колективно.

Проте є декілька особливостей, які можна віднести до недоліків. Це повільність (ця мова програмування хоч і універсальна, проте повільніша за інші), велика кількість ресурсів, необхідних для роботи та «прив'язаність» до системних бібліотек.

Мова програмування Python використовується у наступних сферах:

1. Розробка програмних застосунків будь-якого напрямку.

2. Розробка серверної частини мобільних застосунків (найпопулярніший напрямок).

3. Ігри. Багато сучасних ігор для комп'ютерів (наприклад, World of Tanks) частково чи повністю написані на Python.

4. Вбудовані системи для різних пристроїв. Дуже часто Python використовують для написання внутрішніх платформ управління банкоматами.

5. Скрипти та плагіни до уже реалізованих програм для автоматизації процесів чи створення інших рішень.

6. Тестування (автоматизація цього процесу).

7. Машинне навчання. – основна мова для написання алгоритмів і аналітичних застосунків у сфері Machine Learning.

Бібліотеки Python

Різні бібліотеки Python використовують для виконання конкретних завдань. Наприклад, Matplotlib підходить для відображення даних у двовимірній та тривимірній графіці. Pandas підходить для зручної роботи з даними. NumPy дозволяє створювати масиви та керувати ними. Requests використовується для

веброзробки. OpenCV-Python відкриває можливості для обробки зображень з метою оптимізації систем «машинного зору».

Найвідоміші фреймворки для мови програмування Python

Фреймворки Python допомагають створити зручне та функціональне середовище для розробки. У них міститься набір інструментів, модулів та бібліотек, корисних для виконання конкретних завдань. Це значно полегшує роботу: наприклад, дає змогу не витрачати час на розписування дій, які повторюються, а використати релевантний інструмент. Тож є можливість позбутися рутинних процесів та сконцентруватися на логіці проєкту.

Серед найпопулярніших фреймворків для Python:

– Django – найстаріший та найвідоміший. Створений для реалізації великих інтерактивних проєктів;

– Pyramid – зручний у налаштуваннях, і дає можливість реалізувати складні нестандартні ідеї;

– Web2py – підходить в першу чергу для вебзастосунків і може використовуватись на будь-яких архітектурах.

Популярні Python IDE

IDE або інтегровані середовища розробки – це програмне забезпечення, яке надає розробникам необхідні інструменти для написання, редагування, тестування та налаштування коду. Для розробки на Python найчастіше використовують IDE PyCharm, IDLE, Spyder та Atom.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи відеоспостереження за технологією VSaaS.

В процесі розробки випускної кваліфікаційної роботи за першим

(бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Хмарні рішення швидко опинилися в центрі уваги в багатьох галузях, і безпека відео не є винятком.

VSaaS дозволяє нашим пристроям Edge підключати будь-яку камеру відеоспостереження, щоб наші користувачі могли отримати доступ до своїх IP- або аналогових камер будь-де через нашу хмарну програму.

Оскільки немає потреби додавати додаткове обладнання на додаток до нашого обладнання Edge, VSaaS є ідеальним вибором для малих і середніх компаній із наявним обладнанням, які прагнуть монетизувати традиційні рішення безпеки відео, а також мають можливість додати інтелектуальний штучний інтелект до своїх камер.

Платформа VSaaS AI Edge, будучи незалежною від встановленого обладнання, дозволяє підключати будь-яку камеру, будь то IP або аналогова, і використовувати протоколи виробника для відтворення її функцій, таких як рух PTZ, масштабування, згортання тощо.

У поєднанні з можливістю додавання сенсорних можливостей за допомогою наших додатків штучного інтелекту ви можете надати надзвичайні можливості звичайним камерам, дозволяючи повторно використовувати апаратне забезпечення для наших клієнтів, яким не слід знову інвестувати в апаратне забезпечення.

Віддалений моніторинг граничного обладнання

За допомогою наших хмарних інформаційних панелей ви можете легко контролювати свої команди Edge, де б вони не були встановлені, щоб відповідати вашим контрактам SLA. Деякі з показників, до яких ви також можете отримати доступ:

– Завантаження процесора та використання пам’яті для кожного комп’ютера Edge.

– Статус сховища та доступний дисковий простір.

– Стан мережі, включаючи силу бездротового сигналу та передані дані за типом з’єднання.

– Адміністратори також можуть налаштувати сповіщення про будь-яке з вищезазначеного, щоб вони отримували сповіщення, якщо команда Edge втрачає з’єднання або потребує уваги.

– Ринок додатків відеоаналітики для різних випадків використання.

Усі показники, сповіщення та елементи керування також доступні через API для легкої інтеграції в існуючі інструменти керування мережею та моніторингу.

Cloud Computing і Edge Computing представляють дві різні архітектури. У той час як Cloud Computing обслуговує розподіл послуг через Інтернет, Edge Computing представляє набір інтелектуальних пристроїв, які дозволяють розгортати штучний інтелект і після підключення до мережі дозволяють обмінюватися даними через Інтернет

Камери безпеки обробляють, зберігають і обмінюються все більшою кількістю даних, залежно від роздільної здатності зображення та FPS, отже, об’єднання двох архітектур дозволяє керувати великими обсягами даних, які можна аналізувати для розробки наших рішень штучного інтелекту, застосовуючи аналітику на межі та надсилаючи найважливішу інформацію в хмару, чи то з відеокамер на вулицях, в офісах, у транспортних засобах, а також датчиків IoT.

Екосистема розробника

За допомогою VSaaS API ви можете програмно отримати доступ до даних свого облікового запису для використання в інших програмах. Наприклад, якщо у вашій компанії вже розгорнуто систему збору даних і ви хочете мати доступ до наших даних, ви можете легко зробити це за допомогою нашого API та створити

власні звіти, інформаційні панелі або програми. Або, можливо, ви хочете відобразити наші дані у власному мобільному додатку? Альтернативи безмежні, і ми надаємо будь-яку інформацію:

- Події налаштовано з їх зображеннями, метаданими та правилами.
- Повна конфігурація з налаштованими правилами для кожної камери.
- Усі показники, сповіщення та елементи керування для команд Edge на місцях.
- Метрики, формули чи необроблені дані, будь-що можна проконсультувати.
- Зображення та відео.

3.2 Розробка структурної схеми

Збільшення обсягу та деталізації інформації *(Big Data*), яку збирають підприємства, дає чудову можливість для інтеграторів безпеки, дистриб'юторів і виробників запропонувати нові рішення для запису та зберігання.

Ми говоримо про те, що вже відомо як відеоспостереження як послуга (VSaaS).

Надзвичайне зростання даних у системі відеоспостереження викликало нагальну потребу зберігати, керувати та захищати ці дані з різних причин, або протягом короткого періоду часу, або, у зв'язку з вимогами бізнесу та/або нормативними вимогами, протягом кількох років.

З огляду на таке зростання обсягу відеоданих, придатність використання традиційних відеореєстраторів і відеореєстраторів із локальним сховищем потрібно аналізувати окремо для кожного окремого проекту. Не всі сценарії або користувачі підтримують використання локальних хмарних рішень, тому кожен проект потрібно аналізувати окремо.

У разі вибору рішень для хмарних сховищ, визначити, яку ємність для зберігання найняти, може бути складно, оскільки потреби організації в сховищі залежать від оцінок, що відрізняються різними критеріями.

Незважаючи на те, що багато хто очікує покращення стандартів стиснення, зростання кількості камер, роздільної здатності та часу збереження даних продовжуватиме стимулювати потребу в більш масштабованих і високонадійних рішеннях для зберігання.

Як застосовується VSaaS

У міру розвитку технологій у системах відеоспостереження очікується, що VSaaS та його інфраструктура, розміщена в хмарі, значною мірою сприятимуть зростанню рішень для відеоспостереження. Це може призвести до повного переходу від внутрішніх платформ зберігання до зовнішніх.

Однак перш ніж перейти до хмарних рішень, дуже важливо визначити, чи робить специфіка вашого сценарію технічно можливим і економічно вигідним перехід із локальних рішень для запису на хмарну систему.

Завдяки розміщеній системі компанії можуть використовувати перевірену технологію хмарного зберігання, щоб заощадити кошти без шкоди для продуктивності, ємності чи безпеки. Використання постачальника послуг хмарного зберігання разом із локальним пристроєм зберігання та програмним забезпеченням керування VMS дозволяє користувачам записувати та зберігати відео високої чіткості (HD) локально, забезпечуючи при цьому економічний доступ до відеоданих стандартної роздільної здатності, які безпечно зберігаються в хмарі.

Користувачі можуть віддалено отримувати доступ до файлів високої чіткості з локальних записувачів, а також до даних, розміщених у хмарі, з будь-якого місця, де є підключення до мережі.

Однак важливо зазначити, що існує кілька концепцій хмари: приватна, публічна та змішана. Різниця між ними полягає лише у власниках серверів, на яких здійснюється централізоване зберігання.

Компанії чи установи, які вирішили не використовувати аутсорсинг цього сховища, можуть створити власну приватну хмару у своїх CPD (центрах обробки даних) із серверами високої доступності та рішенням SCATI KRAKEN HA.

Рисунок 3.1 – Структурна схема системи

Переваги VSaaS

Якщо ми зосередимося на кількох конкретних аспектах, рішення для хмарного запису мають ряд переваг порівняно з традиційним локальним записом (DVR/NVR). Однак вони також мають ряд наслідків для IT-відділів, оскільки частина витрат перенаправляється на сферу, яка забезпечує канал зв'язку.

Хоча для відділів безпеки вартість зберігання на одну камеру значно зменшується, а доступність систем практично гарантована 24×7, факт централізованого запису в ЦОД (власний або від зовнішнього постачальника) вимагає, щоб посилання для завантаження та завантаження зображень мали належний розмір, щоб підтримувати весь потік інформації, який проходитиме

через мережу. Також важливо знати, які витрати на передачу як при записі, висхідній лінії зв'язку, так і при перегляді записаних зображень, низхідній лінії зв'язку.

Програмне забезпечення для відеоспостереження або програмне забезпечення CCTV (система замкнутого телебачення) – це засіб безпеки, який допомагає захистити ваш дім чи бізнес від вторгнення та інших ризиків. Він фіксує та записує відеозапис будь-якої діяльності в полі зору, дозволяючи вам стежити за тим, що відбувається в певній області.

Програмне забезпечення для відеоспостереження доступне в багатьох різних формах, включаючи системи IP-камер (також відомі як камери Інтернет-протоколу), NVR (мережеві відеореєстратори) і спеціалізовані комп'ютерні програми, такі як Digital Video Surveillance (DVS). Деякі призначені для використання з існуючими системами безпеки, а інші можуть працювати незалежно.

Переваги встановлення програмного забезпечення для відеоспостереження вдома чи на роботі включають можливість дистанційного моніторингу діяльності. Він також дозволяє швидко й точно виявляти підозрілу поведінку шляхом виявлення руху чи звуку в навколишньому середовищі. Крім того, він надає докази для правоохоронних органів, якщо це необхідно, що може бути корисним під час перевірки незаконних дій, таких як проникнення або крадіжка.

Вибираючи програмне забезпечення для відеоспостереження для вашої системи, ви повинні враховувати такі фактори, як роздільна здатність, частота кадрів (кількість нерухомих зображень, зроблених за секунду), ємність пам'яті (кількість даних, які можна зберегти до того, як їх потрібно буде створити резервну копію), сумісність із мережевими системами (чи буде воно працювати з існуючими IP-камерами чи потребуватиме додаткового обладнання/встановлення) і простота використання – це включає такі функції, як майстри налаштування та зручні інтерфейси.

Для забезпечення оптимальної продуктивності програмне забезпечення для відеоспостереження слід регулярно обслуговувати, виправляючи будь-які вразливості в операційній системі та оновлюючи будь-яке мікропрограмне забезпечення, яке може потребувати заміни через застарілі версії. Ви також захочете переконатися, що у вас є достатньо вільного місця для зберігання на сервері, де зберігаються записи камер відеоспостереження, щоб ви не закінчилися, коли будь-які записи будуть зроблені. Нарешті, важливо не лише захищати права доступу, а й створювати регулярні резервні копії на випадок втрати даних через відключення електроенергії або зловмисну атаку.

Причини використання програмного забезпечення для відеоспостереження

1. **Для моніторингу активності в чутливих зонах:** програмне забезпечення для відеоспостереження можна використовувати, щоб стежити за чутливими зонами та діями, які в них відбуваються. Це можуть бути входи для працівників, склади, банківські сховища, ліфти тощо.

2. **Для покращення внутрішньої безпеки:** за допомогою правильних кутів камери та відеозйомки приміщень служби безпеки можуть швидко ідентифікувати потенційні загрози або підозрілу активність – і все це завдяки програмному забезпеченню для відеоспостереження.

3. **Для стримування потенційних злочинців.** Наявності камер спостереження навколо будівлі часто достатньо, щоб перешкодити злочинцям створювати будь-які неприємності або намагатися вкрасти з даної території. Знаючи, що за ними стежать, зменшується ймовірність того, що вони будуть ризикувати, перебуваючи на місці, захищаючи, у свою чергу, ваші активи.

4. **Для відстеження поведінки працівників:** системи відеоспостереження також дозволяють роботодавцям спостерігати за продуктивністю та поведінкою своїх працівників на роботі. Це допомагає організаціям забезпечити продуктивність, а також надати докази проти будь-яких претензій щодо

невідповідної поведінки чи неналежної поведінки, якщо це необхідно; багато компаній з часом визнали це безцінним.

5. **Для запису інцидентів з метою збору доказів:** доступ до записаного відеозапису інциденту дає правоохоронним органам, страховим компаніям і юристам можливість перевірити, що саме сталося до, під час і після певної події – те, що може стати в нагоді під час пошуку справедливості за протиправні дії чи інші злочини, вчинені особами.

Чому програмне забезпечення для відеоспостереження важливе?

Програмне забезпечення для відеоспостереження є важливим інструментом безпеки. Він допомагає компаніям, установам і урядам захистити свою власність і працівників, а також захищає населення від шкоди. Цей тип програмного забезпечення дозволяє їм відстежувати дії всередині чи за межами своїх приміщень, записувати підозрілу поведінку та зберігати відзнятий матеріал для подальшого аналізу.

Рівень захисту, який пропонує програмне забезпечення для відеоспостереження, може значно відрізнятися залежно від того, яка система використовується. Невеликим підприємствам може знадобитися лише кілька камер навколо їхнього об’єкта з базовими можливостями захоплення зображення, тоді як великим підприємствам можуть знадобитися більш складні системи, які включають виявлення руху, технологію розпізнавання обличчя та функцію повороту та нахилу. Незалежно від розміру чи складності, цей тип програмного забезпечення може забезпечити такий необхідний спокій, коли мова йде про захист персоналу та активів.

Можливість знімати відео в реальному часі також означає, що охоронці можуть швидко реагувати на підозрілу активність або потенційну загрозу. Забезпечуючи миттєве спостереження за будь-якими неочікуваними подіями, що відбуваються на території підприємства або навколо нього, програмне забезпечення для відеоспостереження діє як додатковий набір «очей», які можуть допомогти зупинити злочин до того, як він стався. Крім того, відзнятий матеріал

може слугувати цінним доказом, якщо це буде необхідно у будь-якому кримінальному провадженні.

Окрім переваг фізичного захисту, системи відеоспостереження стають все більш популярними у вирішенні різноманітних проблем, з якими стикаються сучасні організації, починаючи від запобігання втраті запасів і закінчуючи моніторингом продуктивності персоналу. У міру того, як такі інструменти стають все більш досконалими – включають такі аналітичні можливості, як підрахунок людей, які входять і виходять з будівель – вони допомагають користувачам визначати закономірності з часом, які потім дозволяють прийняти розумніші рішення про те, як найкраще керувати своїми ресурсами в майбутньому.

Загалом програмне забезпечення для відеоспостереження стало безцінним надбанням для багатьох різноманітних організацій, допомагаючи як у задоволенні потреб у безпеці, так і сприяючи більшій ефективності та розумінню операцій у цілому.

Функції програмного забезпечення для відеоспостереження

1. **Виявлення руху:** програмне забезпечення для відеоспостереження, як правило, має можливості виявлення руху, що дає змогу виявляти рух у певній зоні та активувати сигнал тривоги чи виконувати інші дії залежно від конфігурації.

2. **Пряма трансляція:** ця функція дозволяє користувачам переглядати в режимі реального часу записи зі своїх камер безпеки з будь-якого пристрою з підключенням до Інтернету. Вони також можуть зберігати записані відео для подальшого перегляду, що робить їх чудовим інструментом як для моніторингу в реальному часі, так і для аналізу після інциденту.

3. **Віддалений доступ:** функції віддаленого доступу дозволяють користувачам дистанційно керувати своїми системами відеоспостереження, наприклад, змінювати ракурси камери або налаштовувати параметри, такі як рівні чутливості та розклади запису, за допомогою веб-браузера або мобільного додатка.

4. **Режим нічного бачення:** деякі системи відеоспостереження оснащені функціями нічного бачення, які дозволяють користувачам контролювати діяльність навіть в умовах слабкого освітлення (наприклад, у приміщенні). Це корисно для приміщень, де джерела світла обмежені або відсутні, щоб люди, яких немає, все ще могли контролювати в будь-який час дня та ночі.

5. **Аналітика та штучний інтелект.** Останні досягнення в галузі штучного інтелекту (ШІ) дозволили багатьом системам відеоспостереження виявляти закономірності, розпізнавати об’єкти та ідентифікувати незвичайну поведінку швидше, ніж будь-коли раніше – і все це без втручання людини. Це робить їх неймовірно потужними інструментами для забезпечення безпеки на підприємствах, у школах, громадських місцях тощо.

Кому може бути корисне програмне забезпечення для відеоспостереження?

– **Власники бізнесу:** програмне забезпечення для відеоспостереження може допомогти власникам бізнесу краще керувати своїми співробітниками та забезпечити підвищену безпеку для клієнтів. Це також може допомогти запобігти збиткам і підвищити загальну ефективність операцій.

– **Правоохоронні органи:** програмне забезпечення для відеоспостереження є потужним інструментом у боротьбі зі злочинністю. Його можливості моніторингу в реальному часі дозволяють офіцерам швидко виявляти підозрілу активність, скорочуючи час реагування на серйозні інциденти.

– **Власники будинків:** системи відеоспостереження за домом є ефективним засобом стримування крадіжок зі зломом та інших вторгнень у житло, забезпечуючи душевний спокій, що ваше майно в безпеці, поки ви відсутні або спите.

– **Школи:** шкільні округи можуть використовувати системи відеоспостереження для спостереження за діяльністю учнів як у шкільних будівлях, так і поза ними, допомагаючи запобігати насильству та вандалізму, створюючи при цьому більш безпечне середовище для учнів.

– **Роздрібні торговці:** роздрібні магазини можуть скористатися перевагами відеоспостереження, щоб зменшити кількість крадіжок, відстежувати транзакції клієнтів, а також захистити працівників від пограбувань або нападів на магазини.

– **Менеджери нерухомості/Орендодавці:** системи відеоспостереження – це чудовий спосіб для орендодавців або менеджерів нерухомості стежити за орендарями без постійної фізичної присутності, що зменшує ризики, пов'язані з безгосподарністю чи зловживанням орендарів.

– **Заклади охорони здоров'я:** лікарні, клініки та будинки престарілих часто використовують системи відеоспостереження, щоб забезпечити безпеку пацієнтів у своїх установах, а також захистити персонал від небажаних відвідувачів або злочинних дій, що відбуваються в медичних центрах.

Скільки коштує програмне забезпечення для відеоспостереження?

Вартість програмного забезпечення для відеоспостереження може істотно відрізнятися в залежності від типу та рівня складності програми. Загалом базове програмне забезпечення для відеоспостереження, яке просто зберігає відзнятий матеріал, може коштувати від 100 до кількох сотень доларів, тоді як більш складні програми з аналітикою, виявленням руху та розпізнаванням обличчя можуть коштувати сотні чи навіть тисячі доларів. Для невеликих бізнес-операцій з обмеженим бюджетом також доступні безкоштовні рішення з відкритим вихідним кодом, які забезпечують базові функції цифрового запису. Крім того, багато компаній вирішують скористатися перевагами хмарних рішень безпеки, вартість яких може варіюватися від 15 до понад 50 доларів на місяць залежно від рівня використання та інших факторів. Зрештою, вартість вашої системи відеоспостереження залежатиме від того, скільки камер вам потрібно та які функції вам потрібні, тому важливо ретельно розглянути всі доступні варіанти, перш ніж приймати будь-які рішення.

Ризики, які слід враховувати з програмним забезпеченням для відеоспостереження

– **Несанкціонований доступ до відеоматеріалів:** зловмисники або зловмисники можуть використовувати потенційні вразливості в системі відеоспостереження, щоб отримати несанкціонований доступ і контролювати відеоматеріали. Це може призвести до зловживання конфіденційними даними, а також призвести до збоїв у роботі служби.

– **Відстеження місцезнаходження та порушення конфіденційності:** системи відеоспостереження можуть дозволяти об'єктам відстежувати пересування та дії людей, що може розглядатися як порушення їхньої конфіденційності.

– **Перешкоди іншим бездротовим пристроям.** Використання бездротових камер може створювати перешкоди іншим пристроям, що працюють у тому ж діапазоні частот, таким як мережі Wi-Fi або пристрої Bluetooth.

– **Хакерські атаки та атаки зловмисного програмного забезпечення:** системи відеоспостереження є вразливими цілями для хакерів через їх підключення до Інтернету, потенційно дозволяючи зловмисникам викрадати конфіденційну інформацію або навіть маніпулювати відзнятим матеріалом для зловмисних намірів. Крім того, застаріле програмне забезпечення може зробити ці системи вразливими до атак шкідливих програм, які можуть порушити роботу служби.

– **Ризик витоку даних:** без належних заходів шифрування незахищені відеоканали можуть бути перехоплені третіми сторонами та використані без дозволу – це може бути особливо проблематично, якщо відзнятий матеріал містить будь-яку конфіденційну інформацію, яку потрібно захистити.

З яким програмним забезпеченням інтегрується програмне забезпечення для відеоспостереження?

Програмне забезпечення для відеоспостереження можна інтегрувати з різними типами програмного забезпечення, залежно від варіанту використання.

Наприклад, можна інтегрувати програмне забезпечення відеоаналітики для аналізу відеопотоків для виявлення руху та розпізнавання обличчя. Системи контролю доступу також можуть бути підключені, щоб дозволити користувачам контролювати, які зони можуть переглядати записи відеоспостереження в режимі реального часу. У системах роздрібної торгівлі системи торгових точок (POS) також можуть бути підключені до програмного забезпечення відеоспостереження, щоб запобігти крадіжкам і контролювати поведінку клієнтів. Інші типи програмного забезпечення, яке можна інтегрувати з відеоспостереженням, включають системи планування ресурсів підприємства (ERP) і системи управління взаємовідносинами з клієнтами (CRM). Нарешті, деякі передові рішення безпеки забезпечують можливості інтеграції з системами автоматизації будівель, такими як контролери HVAC або керування освітленням – це особливо корисно для великих підприємств, яким потрібна складна мережева інфраструктура безпеки.

Питання, які варто поставити, розглядаючи програмне забезпечення для відеоспостереження

1. Який тип камери та технології захоплення відео підтримує програмне забезпечення?

2. Наскільки зручний інтерфейс? Чи знадобиться користувачам навчання для ефективного використання?

3. Чи містить програмне забезпечення такі функції, як виявлення руху, розпізнавання обличчя або аналітика, які можуть запускати тривоги, коли відбуваються заздалегідь визначені події?

4. Який тип пристроїв потрібен для налаштування та яка пропускна здатність потрібна системі для роботи?

5. Наскільки захищена система від витоку даних і які заходи вживаються для захисту переданої інформації, наприклад паролів або потоків?

6. Чи існує максимальне обмеження на час зберігання та витрати, пов’язані зі зберіганням відзнятих матеріалів? Чи доступні резервні копії онлайн у разі втрати пристрою?

7. Чи можна запустити сповіщення, коли виявлено рух або людина проходить повз певні зони в межах дії камер (наприклад, встановлення віртуальних меж)?

8. Які типи тривог можна генерувати, наприклад електронні листи, push-повідомлення, сирени тривоги тощо?

9. Чи можна легко розширити систему відеоспостереження, якщо пізніше знадобляться додаткові камери або датчики?

10. Чи постачається це програмне забезпечення для спостереження з віддаленим доступом, щоб авторизований персонал міг отримати доступ до записів поза межами приміщення (через додаток на мобільних пристроях або через веб-вхід)?

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Систему відеоспостереження за технологією VSaaS реалізуємо функціонально за призначенням, як система відеоспостереження в житлових будинках. Вона вирішує наступні функціональні завдання:

– Цілодобова візуальна оцінка обстановки в житловому секторі шляхом телевізійного спостереження за під'їздами й дворськими територіями, а також усередині будинку за ліфтами й сходовими площадками.

– Оперативне оповіщення служб охорони правопорядку й інших екстрених служб міста про виникнення або підозру на виникнення ситуацій, що загрожують життя й здоров'ю людей, схоронності їхнього майна, а також схоронності муніципального майна.

– Надання службам охорони правопорядку й інших зацікавлених служб міста архівної інформації телевізійного спостереження для відновлення ходу подій, підтримки проведення оперативно-слідчих заходів і т.п.

– Інтеграція систем відеоспостереження з локально-обчислювальними мережами, а також доступ до системи відеоспостереження з будь-якої точки миру, використовуючи підключення до Інтернет.

– Візуальний контроль освітленості під'їздів і дворських територій у темний час доби.

– Візуальний контроль ліфтових кабін, з метою виявлення фактів вандалізму.

– Контроль якості й своєчасності збирання територій, вивозу сміття й т.п.

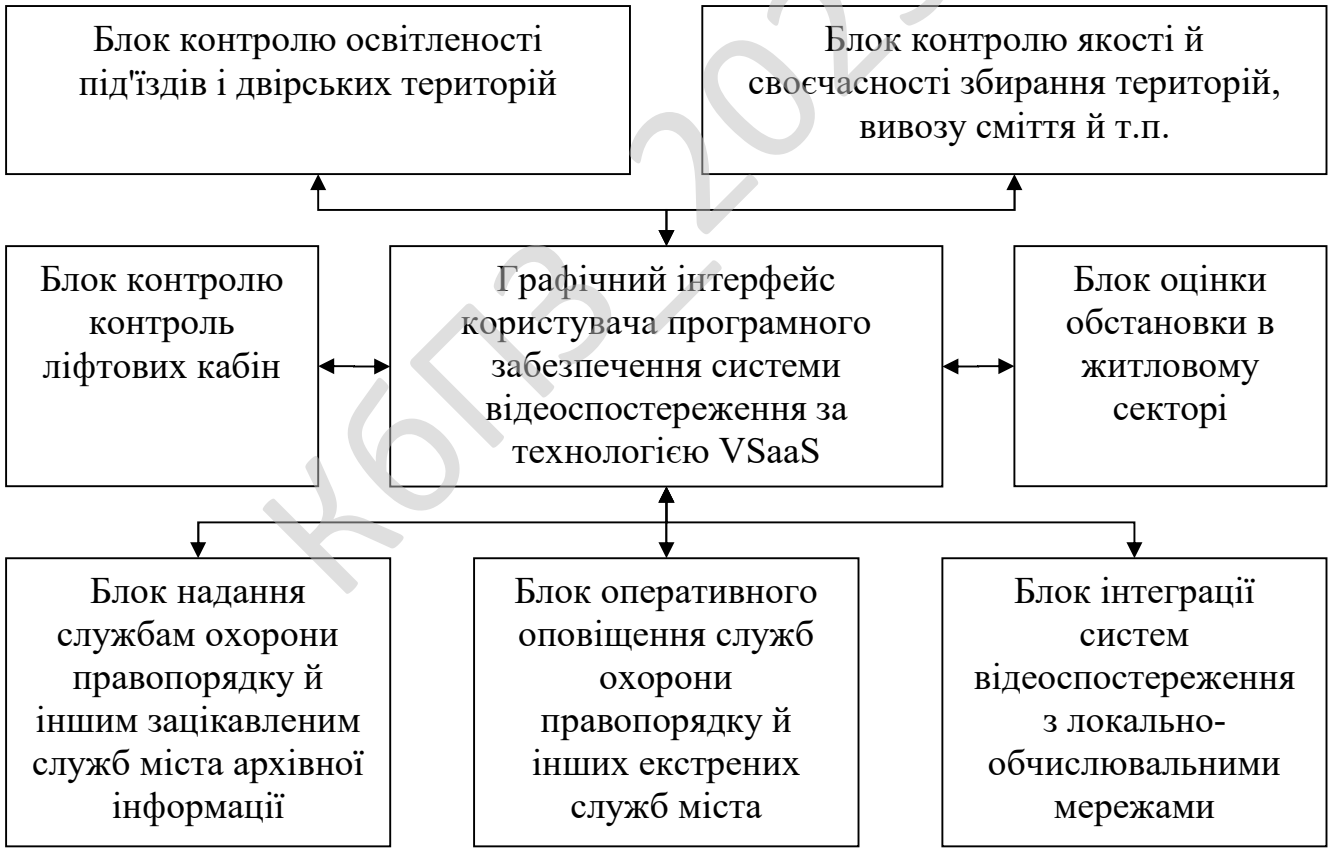

Рисунок 3.2 – Функціональна схема системи

Визначаємо ключові точки відеоспостереження житлового будинку з позиції людей які в ньому проживають:

– відеоспостереження в під'їзді (вхід у під'їзди);

– відеоспостереження стоянки автомобілів;

– відеоспостереження дитячого майданчика.

Тепер розглянемо докладніше кожну зону залежно від завдань системи відеоспостереження будинку.

Відеоспостереження дитячого майданчика

Безпека якої просто неможливо обійти в побудові системи відеоспостереження житлового будинку.

Можливий варіант обійтися однією камерою, але високого дозволу розташувавши її так щоб проглядалася площадка цілком. Дозвіл 700 ТВЛ effio для аналогової камери а у випадку IP відеоспостереження мегапіксельна камера з дозволу й фокусна відстань об'єктива бажано дуже маленьке для широкого кута огляду напр. 2,5мм. 120GRAD.

Аналогова камера (NOVUS NVC-BDN3404H|IR). IP камера відеоспостереження (Mobotix Allround M24 L11) ну дуже широкий кут огляду 180 градусів і дозвіл 3МП.

Відеоспостереження в під'їзді (вхід у парадне)

Особливо важлива насамперед фіксація осіб вхідних/вихідних і людей це камера внутрішня купольна, бажано достатня антивандальність такої камери для запобігання ушкоджень легкодоступного елемента системи відеоспостереження в будинку. Обов'язкова наявність wdr і компенсації задньої засвітки.

Крім камер відеоспостереження для організації системи відеоспостереження будинку знадобляться:

– Кабельна продукція кабель для відеоспостереження – краще використовувати якісний кабель "одесакабель" або європейських виробників.

Зверніть увагу на кількість кабелю для відеоспостереження, якщо купувати відразу бухту 305 м. то в продавця можна одержати знижку, у

відмінності від покупки кабелю по метражу. Відносно кабелю для відеоспостереження використовувати UTP, коаксіальний або кобінований (коаксіальний + ШВВП) залежить у першу чергу від відстані наприклад: 10-70м. Коаксиал, без додаткового встаткування.

– По кручений парі UTP за допомогою приймально-передавачів пасивних або активних передавачів можливо передати відеосигнал до 1000м. Також ураховується ціна, UTP на порядок дешевше коаксіального кабелю для відеоспостереження. На більших відстанях навіть при обліку додаткового встаткування виходить істотна економія. Взагалі якщо відстані від камер відеоспостереження до відеореєстратора великі використовуйте спеціальний відеореєстратор TWIST уже оснащений спеціальним устаткуванням для передачі відеосигналу на більші відстані (до 1000м.), зручним меню, що вражають характеристиками, зроблений в Україні й з нормальним сервісом чого не можна сказати про багатьох неякісних реєстраторах невідомих виробників.

– Коннектори для відеоспостереження, також має сенс використовувати тільки якісні. Якісні коннектори повинні бути як мінімум з міді й мідного сплаву, місце з'єднання коннекторів убереже від впливів зовнішнього середовища монтажна коробка, бажано як мінімум місце контакту помістити в термоусадку.

– Без кабелю також можливо побудувати систему, ір відеоспостереження запросто, навіть у випадку, камер без wi-fi модуля, установкою додаткового мережного бездротового встаткування (wi-fi точка доступу для відеонагляду, wi-fi роутер)

– Блок живлення для системи відеоспостереження один або декілька, розраховується відповідно до споживаних потужностей камер, але з деяким запасом на втрату живлення в проводах або пікові перегони при включенні ІЧ підсвічування, наприклад якщо камери споживають 2,8 А, те вибираючи між блоком живлення 3А и 5 А варто зупинити вибір на 5А.

– Реєстратор для системи відеоспостереження багатоквартирного будинку необхідний із частотою 25 кадрів в d1 25к/с як мінімум на каналах дитячого майданчика й вхід у парадне, там найбільша прохідність.

– А у випадку IP відеоспостереження відповідально підійти до вибору ПЗ й модулів, що підключаються, наприклад модуль для фіксації номерів автотранспорту, або модуль визначення осіб – ідентифікація відвідувачів.

Також бажано вирішити питання з віддаленим доглядом для мешканців непогано настроїти доступ для віддаленого перегляду через хмарний сервіс для відеоспостереження.

Немаловажно при установці відеоспостереження будинку відразу вирішити питання про сервісне обслуговування цієї системи. Або це буде разовий візит фахівця у випадку виходу з ладу, або ж більше кращий варіант укласти договір сервісного обслуговування. У цьому випадку оплата буде щомісяця або щокварталу, але крім разового виїзду при поломці системи одержите профілактичні огляд, настроювання при необхідність і пріоритет в обслуговуванні в порівнянні з разовим виїздом на ремонт відеоспостереження.

Відеоспостереження стоянки автомобілів

Загальний контроль за стоянкою й контроль що в'їжджає й виїжджає автотранспорту.

Необхідна камера вуличного виконання з повноцінним режимом день/ніч або ІЧ підсвічуванням. Бажана наявність у камері функції антивідблиск, для запобігання засвітки камери автомобільними фарами. Як варіант на в'їзді на стоянку камера для фіксації автомобільних номерів, особливо зручно якщо в'їзд на стоянку обмежений за допомогою шлагбаума, або у випадку використання спеціального програмного забезпечення для відеоспостереження тоді спрощується облік що заїхали, що виїхали автомобілів, до речі стаття про відеоспостереження на транспорті, Тобто на стоянці розташовані як мінімум дві камери відеоспостереження одна для загального контролю відеоспостереження на стоянці, а друга для фіксації державних номерів що в'їжджають і виїжджають

автомобілів. Які переваги житлового будинку із системою безпеки й без такої, пояснювати по моєму не треба це насамперед безпека людей які там живуть. Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється. Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

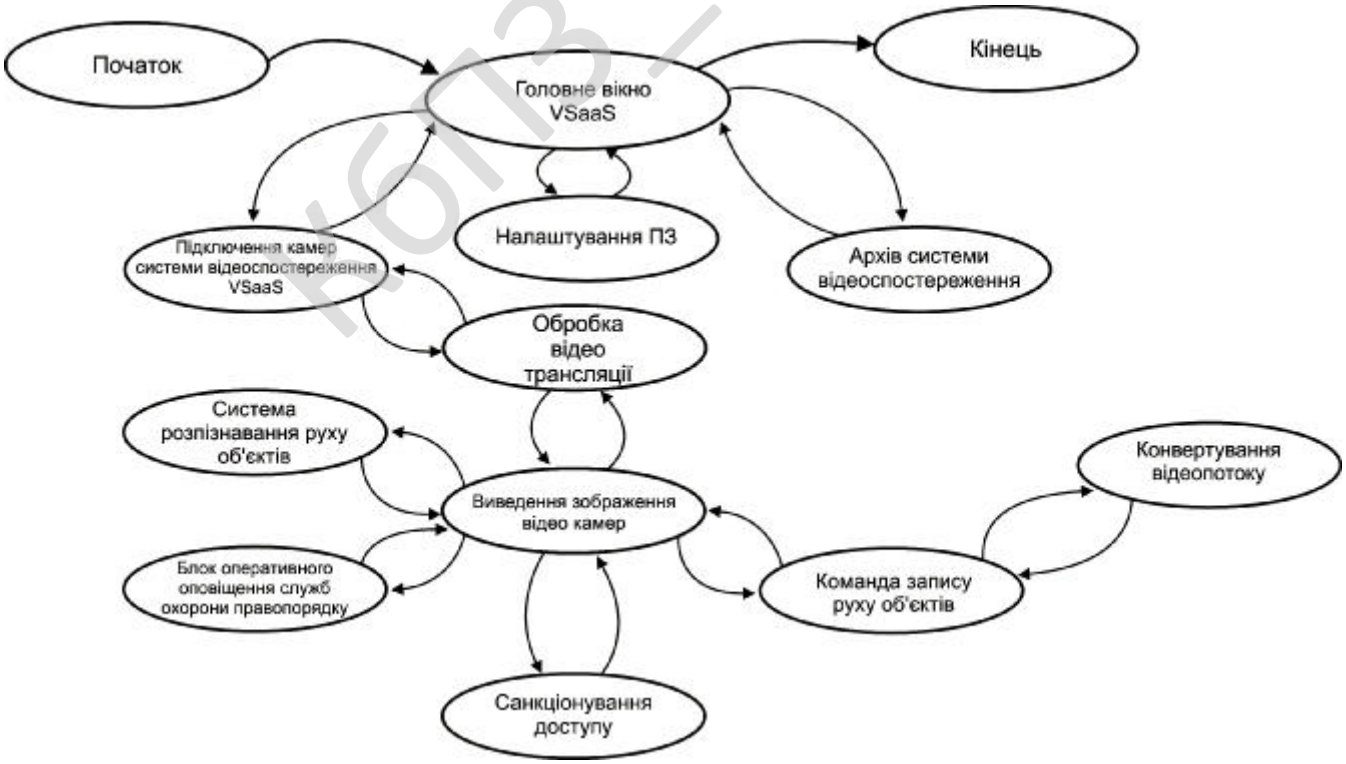

Рисунок 3.3 – Діаграма взаємодії процесів

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Блок-схеми є основою ПЗ. Тому від точності і детальності проробки блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації, також те, що при розробці програми слід надати особливу увагу модулю відеоспостереження за технологією VSaaS.

Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні блоки можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірки поточного стану та поверненням на початок схеми чи з завершенням роботи розробленого ПЗ.

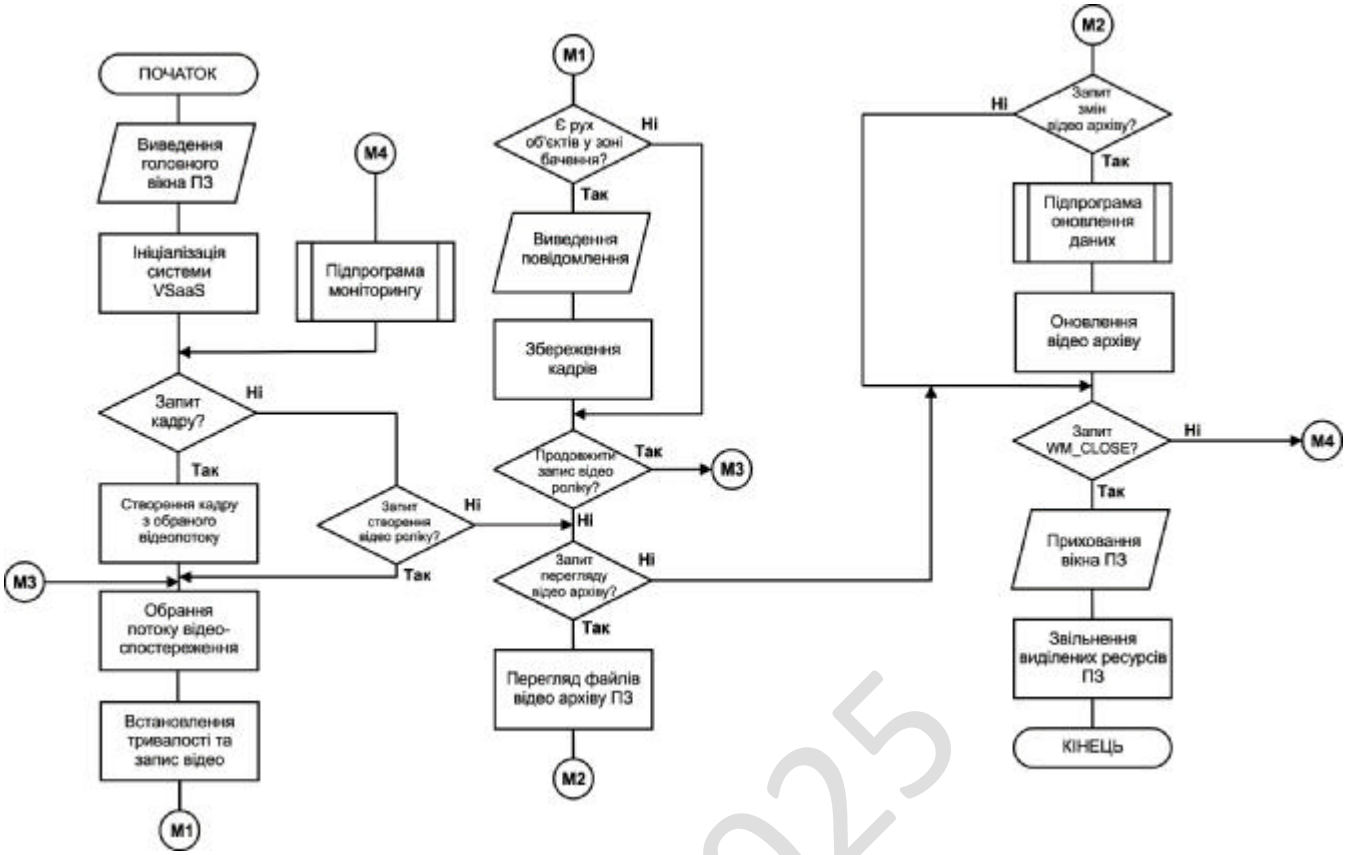

Рисунок 4.1 – Блок-схема основної програми

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

Розглянемо використані технології та їх основні компоненти що підтверджують правильність використаних проектних рішень.

При розробці ПЗ було використано підходи ри́зик-ме́неджменту – це система управління ризиками, яка включає в себе стратегію та тактику управління, направлені на досягнення основних цілей. Ефективний ризик-менеджмент включає:

– систему управління;

– систему ідентифікації і вимірювання;

– систему супроводження (моніторингу та контролю).

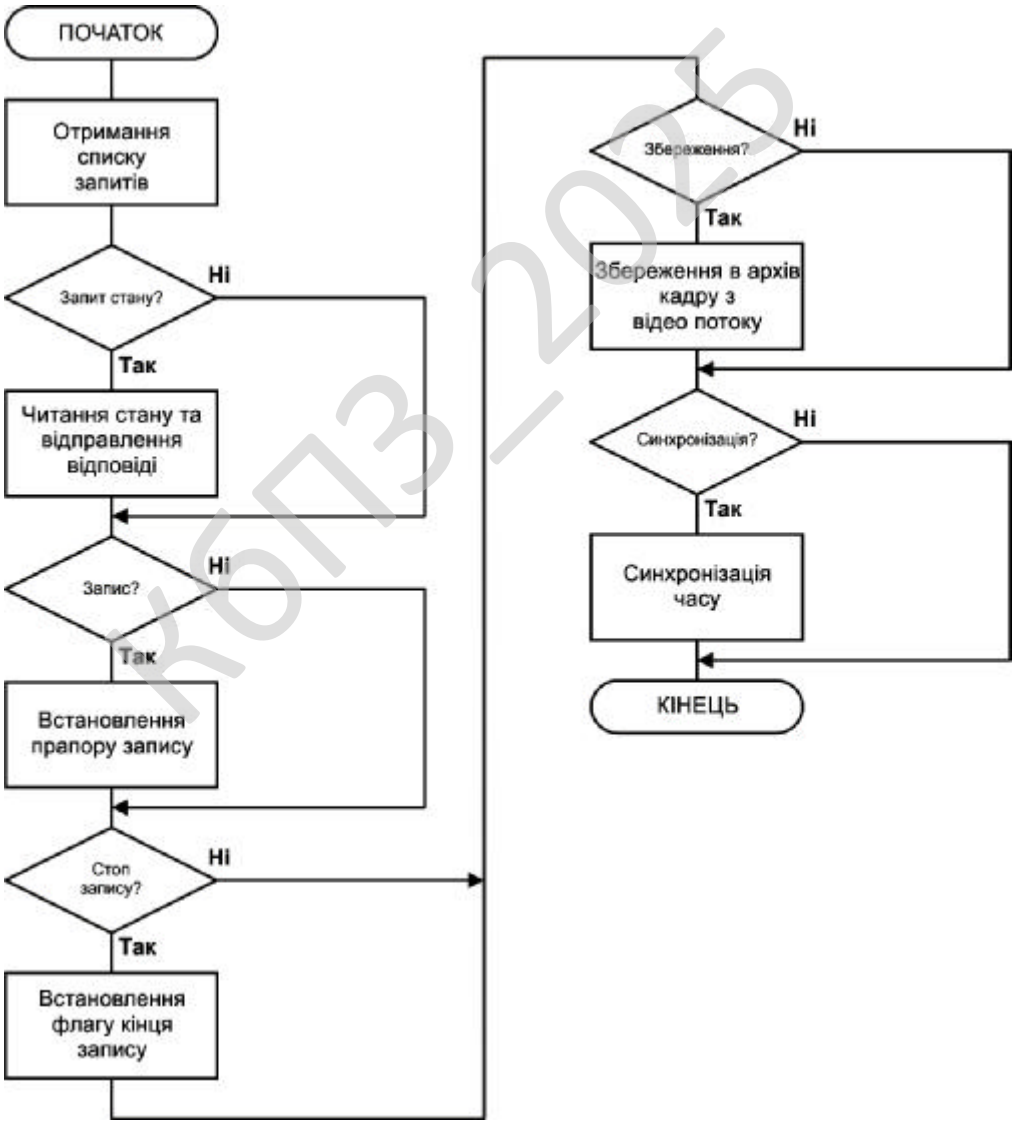

Рисунок 4.2 – Блок-схема роботи підпрограми

Сучасна наука представляє ризик як вірогідну подію, в результаті настання якої можуть відбутися позитивні, нейтральні або негативні наслідки. Якщо ризик припускає наявність як позитивних, так і негативних результатів, він відноситься до спекулятивних ризиків. Якщо ж наслідки негативні, або відсутні взагалі, такий ризик іменується чистим.

Мета ризик-менеджменту – підвищення конкурентоспроможності господарюючих суб'єктів за допомогою захисту від реалізації чистих ризиків.

Теорія ризик-менеджменту ґрунтується на трьох базових поняттях: корисності, регресії і диверсифікації.

У 1738 швейцарський математик Даніель Бернуллі доповнив теорію вірогідності методом корисності або привабливості того або іншого результату подій. Ідея Бернуллі полягала в тому, що в процесі ухвалення рішення люди приділяють більше уваги розміру наслідків різних результатів, ніж їх вірогідність.

В кінці XIX століття англійський дослідник Ф. Гальтон запропонував вважати регресію або повернення до середнього значення універсальною статистичною закономірністю. Суть регресії трактувалася ним як повернення явищ до норми з часом. Згодом було доведено, що правило регресії діє в найрізноманітніших ситуаціях, починаючи з азартних ігор та розрахунку вірогідності виникнення нещасних випадків, і закінчуючи прогнозуванням коливань економічних циклів.

У 1952 аспірант Університету Чикаго Гарі Марковіц в статті «Диверсифікація вкладень» («Portfolio Selection») математично обґрунтував стратегію диверсифікації інвестиційного портфеля, зокрема, він показав, як шляхом продуманого розподілу вкладень мінімізувати відхилення прибутковості від очікуваного показника. У 1990 Г. Марковіцу присуджена Нобелівська премія за розробку теорії і практики оптимізації портфеля фондових активів.

Етапи ризик-менеджменту

У ризик-менеджменті прийнято виділяти декілька ключових етапів:

– на першому етапі відбувається виявлення ризику з супутньою оцінкою вірогідності його реалізації і масштабу наслідків;

– на другому етапі здійснюється розробка ризик-стратегії з метою зниження вірогідності реалізації ризику і мінімізації можливих негативних наслідків;

– на третьому етапі вибираються методи і інструменти управління виявленим ризиком;

– на четвертому етапі проводиться безпосереднє управління ризиком;

– на завершальному етапі оцінюються досягнуті результати і коректується ризик-стратегія.

За ключовий етап ризик-менеджменту вважається етап вибору методів і інструментів управління ризиком.

Методи і інструментарій ризик-менеджменту

Базовими методами ризик-менеджменту є відмова від ризиків, зниження, передача і ухвалення.

Ризик-інструментарій значно ширший. Він включає політичні, організаційні, правові, економічні, соціальні інструменти, причому ризик-менеджмент як система допускає можливість одночасного застосування декількох методів і інструментів ризик-управління.

Найбільш часто вживаним інструментом ризик-менеджменту є страхування. Страхування припускає передачу відповідальності за відшкодування передбачуваного збитку сторонній організації (страхової компанії).

Прикладами інших інструментів можуть бути відмова від надмірно ризикової діяльності (метод відмови), профілактика або диверсифікація (метод зниження), аутсорсинг витратних ризикових функцій (метод передачі), формування резервів або запасів (метод ухвалення).

Система відеоспостереження за технологією VSaaS є сучасним рішенням для віддаленого моніторингу та аналізу відеопотоків. Вона використовує хмарні сервіси для зберігання та обробки відео, забезпечуючи доступність у режимі реального часу. Реалізація системи здійснюється мовою програмування Python.

Архітектура системи

Система складається з декількох ключових компонентів. Перший компонент – відеокамери, що здійснюють запис відеопотоку та передають дані на сервер. Другий компонент – серверна частина, що включає модулі обробки, аналізу та зберігання відео. Третій компонент – клієнтський інтерфейс, через який користувач отримує доступ до відеопотоків та архівів.

Функціональні можливості

Система виконує декілька основних функцій. Вона здійснює запис та трансляцію відео у режимі реального часу. Дані передаються через захищений канал зв'язку. Система зберігає відео у хмарному сховищі. Передбачено функцію виявлення руху та ідентифікації об'єктів за допомогою алгоритмів комп'ютерного зору. Доступ користувачів до відеопотоків контролюється системою автентифікації.

Основні модулі системи

Серверна частина включає декілька модулів. Модуль обробки відео відповідає за прийом потоків, їх обробку та архівацію. Модуль детекції руху використовує бібліотеки OpenCV та TensorFlow для аналізу кадрів та виявлення змін у зображенні. Модуль управління користувачами містить систему авторизації та доступу.

Клієнтська частина реалізована за допомогою веб-інтерфейсу. Вона дозволяє переглядати відеопотоки у реальному часі, здійснювати пошук у архіві та налаштовувати параметри камер.

Вихідний код та використані бібліотеки

Для роботи з відеопотоками використовується бібліотека OpenCV. Передача відео здійснюється через WebRTC. Хмарне зберігання реалізується за

допомогою Amazon S3 або Google Cloud Storage. Серверна частина працює на Flask, що забезпечує взаємодію клієнта з сервером через REST API.

```
import cv2
import numpy as np
from flask import Flask, Response
app = Flask(__name__)
camera = cv2.VideoCapture(0)
def generate_frames():
    while True:
        success, frame = camera.read()
        if not success:
            break
        else:
            _, buffer = cv2.imencode('.jpg', frame)
            frame_bytes = buffer.tobytes()
            yield (b'--frame\r\n'
b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n')
@app.route('/video')
def video_feed():
    return Response(generate_frames(),
                    mimetype='multipart/x-mixed-replace; boundary=frame')
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

У цьому коді здійснюється отримання відеопотоку з камери та передача його через HTTP. Використовується Flask для створення веб-сервера та OpenCV для обробки зображень.

Розрахунки продуктивності системи

Система має обмеження щодо обсягу переданих даних. Наприклад, якщо одна камера передає відео у роздільній здатності 1920x1080 пікселів з частотою 30 кадрів за секунду, а кодек H.264 стискає потік до 4 Мбіт/с, то при наявності 10 камер необхідний канал зв’язку мінімум 40 Мбіт/с. При використанні 10 камер обсяг даних на добу становить 432 ГБ.

Для зберігання відео за добу для однієї камери потрібно:

– 4 Мбіт/с = 0.5 МБ/с.

– 0.5 МБ/с × 3600 с = 1.8 ГБ/год.

– 1.8 ГБ/год × 24 = 43.2 ГБ/доба.

Розрахунки підтверджують, що для ефективної роботи системи необхідно використовувати хмарне зберігання з високою пропускною здатністю каналу передачі даних. Оптимізація трафіку досягається застосуванням кодека H.264 та технології детекції руху, що дозволяє записувати лише зміни у кадрі. Використання Python та OpenCV дозволяє реалізувати систему з мінімальними витратами ресурсів, а Flask забезпечує доступність відеопотоків через веб-інтерфейс.

Механізми безпеки (шифрування відеопотоків та контроль доступу)

Для безпечної передачі відеопотоків через HTTP використовуємо SSL/TLS. Flask може працювати з SSL-сертифікатом, а передача даних шифрується.

Також реалізується контроль доступу через авторизацію користувачів за токенами (JWT).

Як працює ця система безпеки

1. Користувач авторизується за допомогою /login, вводячи логін та пароль.

2. Якщо логін і пароль правильні, сервер повертає JWT токен.

3. Для доступу до відеопотоку користувач повинен надсилати цей токен у заголовку запиту.

4. Сервер перевіряє токен перед тим, як надати доступ до /video.

5. Передача відео відбувається через SSL-з’єднання.

Реалізація серверної частини з захищеним відеопотоком:

```
import cv2
import numpy as np
import jwt
import datetime
from flask import Flask, Response, request, jsonify
app = Flask(__name__)
app.config['SECRET_KEY'] = 'supersecretkey'
# Секретний ключ для підпису токенів
camera = cv2.VideoCapture(0)
```

```
# Функція для генерації JWT токена
def generate_token(username):
    payload = {
        'user': username,
        'exp': datetime.datetime.utcnow() + datetime.timedelta(hours=1)
    }
    return jwt.encode(payload, app.config['SECRET_KEY'], algorithm="HS256")

# Перевірка JWT токена
def token_required(func):
    def wrapper(*args, **kwargs):
        token = request.headers.get('Authorization')
        if not token:
            return jsonify({'message': 'Токен відсутній'}), 403
        try:
jwt.decode(token.split()[1],
           app.config['SECRET_KEY'], algorithms=["HS256"])
        except jwt.ExpiredSignatureError:
            return jsonify({'message': 'Токен закінчив дію'}), 403
        except jwt.InvalidTokenError:
            return jsonify({'message': 'Недійсний токен'}), 403
        return func(*args, **kwargs)
    return wrapper

# Головна сторінка
@app.route('/')
def home():
    return "Система відеоспостереження VSaaS"
# Генерація відеопотоку
def generate_frames():
    while True:
        success, frame = camera.read()
        if not success:
            break
        else:
            _, buffer = cv2.imencode('.jpg', frame)
            frame_bytes = buffer.tobytes()
            yield (b'--frame\r\n'
          b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n')
# Захищений маршрут для перегляду відеопотоку
@app.route('/video')
@token_required
```

```
def video_feed():
    return Response(generate_frames(),
                    mimetype='multipart/x-mixed-replace; boundary=frame')

# Авторизація та отримання токена
@app.route('/login', methods=['POST'])
def login():
    data = request.get_json()
    if not data or 'username' not in data or 'password' not in data:
        return jsonify({'message': 'Некоректні дані'}), 400

    # Простий приклад перевірки користувача (можна замінити на базу даних)
    if data['username'] == 'admin' and data['password'] == 'password':
        token = generate_token(data['username'])
        return jsonify({'token': token})
    else:
        return jsonify({'message': 'Невірний логін або пароль'}), 401

if __name__ == '__main__':
    app.run(ssl_context=('cert.pem', 'key.pem'), host='0.0.0.0', port=5000)
```

Виявлення руху (Motion Detection)

Як працює ця система:

1. Зчитується перший кадр (еталонний), який буде використовуватися для порівняння.

2. Кожен новий кадр порівнюється з еталонним, щоб визначити зміни.

3. Якщо зміни перевищують встановлений поріг, система розпізнає це як рух.

4. Виводиться зображення з позначеними контурами рухомих об’єктів.

Цей алгоритм аналізує зміни між кадрами, використовуючи OpenCV.

```
import cv2
import numpy as np
camera = cv2.VideoCapture(0)
# Зчитування першого кадру для порівняння
_, first_frame = camera.read()
first_gray = cv2.cvtColor(first_frame, cv2.COLOR_BGR2GRAY)
first_gray = cv2.GaussianBlur(first_gray, (21, 21), 0)
```

```
        while True:
            _, frame = camera.read()
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            gray = cv2.GaussianBlur(gray, (21, 21), 0)

            # Визначення змін між кадрами
            delta = cv2.absdiff(first_gray, gray)
            threshold = cv2.threshold(delta, 25, 255, cv2.THRESH_BINARY)[1]
            threshold = cv2.dilate(threshold, None, iterations=2)

            # Контури знайдених рухомих об’єктів
            contours,    _    =    cv2.findContours(threshold,    cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
            for contour in contours:
                if cv2.contourArea(contour) < 1000:
                    continue
                x, y, w, h = cv2.boundingRect(contour)
                cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            cv2.imshow("Motion Detection", frame)
            if cv2.waitKey(1) == ord("q"):
                break
        camera.release()
        cv2.destroyAllWindows()
```

Алгоритми машинної графіки можна розділити на два рівні: нижній і верхній. Група алгоритмів нижнього рівня призначена для реалізації графічних примітивів (математичних розрахунків, ліній, кіл, заповнень і т.п.).

Ці алгоритми або подібні їм відтворені в графічних бібліотеках мов високого рівня або реалізовані апаратний в графічних процесорах робочих станцій.

Серед алгоритмів нижнього рівня можна виділити наступні групи.

1. Найпростіші в значенні математичних методів і відмінні простотою реалізації, що використовуються. Як правило, такі алгоритми не є як найкращими за об'ємом виконуваних обчислень або необхідним ресурсам пам'яті.

2. Алгоритми з складнішими математичними передумовами, що відрізняють їх більшу ефективність.

3. До третьої групи слід віднести алгоритми, які можуть бути без великих ускладнень реалізовані апаратний (допускаючі розпаралелювання, рекурсивні, реалізовувані в найпростіших командах). В цю групу можуть потрапити і алгоритми, представлені в перших двох групах.

4. Нарешті, до четвертої групи можна віднести алгоритми із спеціальним призначенням (наприклад, для усунення сходового ефекту, тремтіння кадру та інші).

До алгоритмів верхнього рівня відносяться в першу чергу алгоритми видалення невидимих ліній і поверхонь.

Задача видалення невидимих ліній і поверхонь продовжує залишатися центральною в машинній графіці..

До задачі видалення невидимих ліній і поверхонь примикає задача побудови (замальовування) півтонових (реалістичних) зображень, тобто обліку явищ, пов'язаних з кількістю і характером джерел світла, обліку властивостей поверхні тіла (прозорість, заломлення, віддзеркалення світла).

Проте, при цьому не слід забувати, що виведення об'єктів в алгоритмах верхнього рівня забезпечується примітивами, що реалізовують алгоритми нижнього рівня, тому не можна ігнорувати проблему вибору і розробки ефективних алгоритмів нижнього рівня.

Для різних областей застосування машинної графіки на перший план можуть висуватися різні властивості алгоритмів.

Для наукової графіки велике значення має універсальність алгоритму, при цьому швидкодія може відходити на другий план. Для систем моделювання, відтворюючих об'єкти, що рухаються, швидкодія стає головним критерієм, оскільки потрібно генерувати зображення практично в реальному масштабі часу.

Особливості растрової графіки зв'язані з тим, що звичайні зображення, з якими стикається людина в своїй діяльності (креслення, графіки, карти, художні картини і т.п.), реалізовані на площині, що складається з нескінченного набору крапок.

Екран же растрового дисплея представляється матрицею дискретних елементів, що мають конкретні фізичні розміри.

При цьому число їх істотно обмежено. Тому не можна провести точну лінію з однієї крапки в іншу, а можна виконати тільки апроксимацію цієї лінії з відображенням її на дискретній матриці (площини).

Таку площину також називають цільна чисельними гратами, растровою площиною або растром. Ці грати представляються квадратною сіткою з кроком 1. Відображення будь-якого об'єкту на цільна чисельні грати називається розкладанням його в растр або просто растровим уявленням.

Архітектура ПЗ що використовувалась

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних програм і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

– набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;

– набір клієнтів, які використовують сервіси, що надаються серверами;

– мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

Дуже важливо ясно уявляти, хто або що розглядається як «клієнт». Можна говорити про клієнтський комп'ютер, з якого відбувається звернення до інших комп'ютерів. Можна говорити про клієнтське та серверне програмне забезпечення. Нарешті, можна говорити про людей, які бажають за допомогою

відповідного програмного та апаратного забезпечення отримати доступ до тієї чи іншої інформації.

Загальноприйнятим є положення, що клієнти та сервери – це перш за все програмні модулі. Найчастіше вони знаходяться на різних комп'ютерах, але бувають ситуації, коли обидві програми – і клієнтська, і серверна, фізично розміщуються на одній машині; в такій ситуації сервер часто називається локальним.

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

– рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;

– прикладний рівень, який реалізує основну логіку ПЗ і на якому здійснюється необхідна обробка інформації;

– рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дворівнева клієнт-серверна архітектура передбачає взаємодію двох програмних модулів – клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

– модель тонкого клієнта, в рамках якої вся логіка ПЗ та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;

– модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін.

Типовим прикладом клієнт-серверної взаємодії є WWW. Існує величезна кількість веб-серверів, на яких розміщується та чи інша інформація. У

найпростішому випадку ця інформація являє собою набір веб-сторінок, які можуть зберігатися на сервері у вигляді файлів, розмічених за допомогою мови розмітки HTML. Але ситуація, як правило, є складнішою; значна частина веб-ресурсів на сучасному етапі є динамічними, тобто вони не існують в заздалегідь підготовленому вигляді, а створюються безпосередньо в процесі обробки запиту від користувача.

Для того, щоб людина, яка працює в Інтернеті, могла переглянути ту чи іншу сторінку, на її комп'ютері повинно бути встановлено відповідне програмне забезпечення. Програми для перегляду веб-сторінок називаються браузерами.

Але, крім браузерів, до серверів можуть звертатися і інші клієнти, а саме – автономні програми. Вони можуть передбачати взаємодію з людиною, а можуть працювати в цілком автоматичному режимі. Типовим класом таких програм є роботи, призначені для автоматичного перегляду веб-ресурсів. Зокрема, роботи є важливим елементом пошукових систем і використовуються ними для перегляду сторінок і збору інформації про них.

Для запиту до веб-сервера клієнтська програма повинна задати місцезнаходження комп'ютера, на якому розміщується серверна програма, назву потрібного документа і, можливо, інші дані, які специфікують запит. Мережа забезпечує знаходження сервера і передачу йому клієнтського запиту. Серверні програми обробляють цей запит, відповідь пересилається по мережі клієнтові.

Трирівнева клієнт-серверна архітектура, яка почала розвиватися з середини 90-х років, передбачає відділення прикладного рівня від управління даними. Відокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка ПЗ. Програми проміжного рівня можуть функціонувати під управлінням спеціальних серверів ПЗ, але запуск таких програм може здійснюватися і під управлінням звичайного веб-сервера. Нарешті, управління даними здійснюється сервером даних.

Для роботи з системою користувач використовує стандартне програмне забезпечення –звичайний браузер. Це позбавляє його необхідності завантажувати

та інсталювати спеціальні програми (хоча інколи така необхідність все-таки виникає).

Але користувачеві слід надати в розпорядженні інтерфейс, який дозволяв би йому взаємодіяти з системою і формувати запити до неї. Форми, що визначають цей інтерфейс, розміщуються на веб-сторінках та завантажуються разом з ними.

Веб-оглядач формує запит та пересилає його до сервера, який здійснює обробку. При необхідності сервер викликає серверні програмні модулі, які забезпечують обробку запиту і в разі потреби звертаються до сервера даних. Сервер даних здійснює операції з даними, що зберігаються в системі та складають її інформаційну основу. Зокрема, він може здійснити вибірку з інформаційної бази відповідно до запиту та передати її модулю проміжного рівня для подальшої обробки. Дані, з якими працює сервер даних, найчастіше організовані як реляційна база даних.

Найчастіше веб-сервер і серверні модулі проміжного рівня розміщуються на одному комп'ютері, хоч і являють собою окремі і логічно незалежні програмні модулі.

На сучасному етапі для програмування модулів проміжного рівня використовується мова серверних сценаріїв PHP, а для управління даними – СУБД MySQL. Таким чином, зв'язку PHP-MySQL слід розглядати як стандартний інструмент для створення порівняно простих інтерактивних веб-сайтів та систем електронної комерції; близько 90% комерційних систем сьогодні створюється саме на цій основі. Водночас як засоби управління даними, так і middleware-засоби можуть бути найрізноманітнішими. Так, для створення серверних програм, крім PHP, широко застосовуються Java, Perl, Python, Delphi.

Взагалі, технології створення розподілених, зокрема веб-програм, стрімко розвиваються. Слід згадати про технології EJB (Enterprise Java Beans), CORBA, а також про .NET – порівняно нову ініціативу компанії Microsoft. Для зберігання даних та їх передачі часто використовується так звана розширювана мова

розмітки XML (Extensible Markup Language).

Використана методологія розробки

Незважаючи на те що я працював над ПЗ один в реалізації програми я використовував підходи пришвидшення розробки на основі методологій Agile.

Гнучка́ розробка програмного забезпечення (Agile software development, agile-методи) – клас методологій розробки програмного забезпечення, що базується на ітеративній розробці, в якій вимоги та розв'язки еволюціонують через співпрацю між самоорганізовуваними багатофункціональними командами.

Гнучка розробка – найкращий засіб для підвищення продуктивності розробників програмного забезпечення.

Більшість гнучких методологій націлені на мінімізацію ризиків, шляхом зведення розробки до серії коротких циклів, що мають назву ітерацій, які зазвичай тривають один-два тижні. Кожна ітерація сама по собі виглядає як програмний проект в мініатюрі, і включає всі завдання, необхідні для видачі мінімального приросту за функціональністю: планування, аналіз вимог, проектування, кодування, тестування і документування. Хоча окрема ітерація, як правило, недостатня для випуску нової версії продукту, мається на увазі те, що гнучкий програмний проект готовий до випуску наприкінці кожної ітерації. Після закінчення кожної ітерації, команда виконує переоцінку пріоритетів розробки.

Agile акцентує увагу на безпосередньому спілкуванні «віч-на-віч». Більшість agile команд розташовані в одному офісі, його іноді називають bullpen. Як мінімум вона включає і «замовників» (замовники, які визначають продукт, також це можуть бути менеджери продукту, бізнес аналітики або клієнти). Офіс може також включати тестувальників, дизайнерів інтерфейсу, технічних авторів і менеджерів.

Основною метрикою agile методів є робочий продукт. Віддаючи перевагу безпосередньому спілкуванню, agile-методи зменшують обсяг письмової документації в порівнянні з іншими методами. Це привело до критики цих методів як недисциплінованих.

Agile – родина процесів розробки, а не єдиний підхід в розробці програмного забезпечення, і визначається Agile Manifesto. Agile не включає практик, а визначає цінності та принципи, якими керуються успішні команди.

Agile Manifesto розроблений і прийнятий 17 розробниками 11-13 лютого 2001 року на лижному курорті The Lodge at Snowbird в горах Юти. Маніфест підписали представники наступних методологій Extreme programming, Scrum, DSDM, Adaptive software development, Crystal Clear, Feature driven development, Pragmatic Programming. Agile Manifesto містить 4 основні ідеї та 12 принципів. Примітно, що Agile Manifesto не містить практичних порад.

Основні ідеї:

– Особистості та їхні взаємодії важливіші, ніж процеси та інструменти;

– Робоче програмне забезпечення важливіше, ніж повна документація;

– Співпраця із замовником важливіша, ніж контрактні зобов'язання;

– Реакція на зміни важливіша, ніж дотримання плану.

Принципи, які роз'яснює Agile Manifesto:

– задоволення клієнта за рахунок ранньої та безперебійної поставки коштовного програмного забезпечення;

– вітання змін вимог навіть наприкінці розробки (це може підвищити конкурентоспроможність отриманого продукту);

– часта поставка робочого програмного забезпечення (кожен місяць або тиждень або ще частіше);

– тісне, щоденне спілкування замовника з розробниками впродовж всього проекту;

– проектом займаються мотивовані особистості, які забезпечені потрібними умовами роботи, підтримкою і довірою;

– рекомендований метод передачі інформації – особиста розмова (віч-на-віч);

– робоче програмне забезпечення – найкращий вимірювач прогресу;

– спонсори, розробники та користувачі повинні мати можливість підтримувати постійний темп на невизначений термін;

– постійну увагу поліпшенню технічної майстерності та зручному дизайну;

– простота – мистецтво не робити зайвої роботи;

– найкращі технічні вимоги, дизайн та архітектура виходять у самоорганізованої команди;

– постійна адаптація до мінливих обставин.

Маніфест та Принципи гнучкої розробки містять високорівневі ідеї щодо того, як потрібно вибудовувати процес розробки програмного забезпечення, щоб успішно завершувати проекти й створювати команди, в яких приємно та цікаво працювати.

Документи визначають, що потрібно для цього зробити, але не говорять, як це зробити. По-іншому й не могло бути, оскільки Маніфест та Принципи народилися внаслідок консенсусу представників різних (хоча й споріднених) напрямів, які могли знайти спільну основу лише на рівні базових цінностей та принципів.

Критика. Багато керівників проектів, що працюють у традиційних методологіях на кшталт «водоспаду», критикують agile-методи.

Один з повторюваних пунктів критики: при agile-підході часто нехтують створенням «дорожньої карти» розвитку продукту, так само як і управлінням вимогами, в процесі якого і формується така «карта». Гнучкий підхід до управління вимогами не має на увазі далекосяжних планів (по суті, управління вимогами просто не існує в даній методології), а має на увазі можливість замовника раптом і несподівано наприкінці кожної ітерації виставляти нові вимоги, що часто суперечать архітектурі вже створеного і поставленого продукту. Таке іноді призводить до катастрофічних «авралів» з масовим рефакторингом і переробками практично на кожній черговій ітерації.

Крім того вважається, що робота в agile мотивує розробників вирішувати всі прибулі завдання найпростішим і найшвидшим можливим способом, при

цьому часто не звертаючи уваги на коректність коду з точки зору вимог базової платформи (підхід «працює, та й добре», при цьому не враховується, що може перестати працювати при найменшій зміні або ж породити важкі до відтворення дефекти після реального розгортання у клієнта). Це призводить до зниження якості продукту і накопиченню дефектів.

Методології. Існують методології, які дотримуються цінностей і принципів заявлених в Agile Manifesto, деякі з них:

1. Agile Modeling – набір понять, принципів і прийомів (практик), що дозволяють швидко і просто виконувати моделювання і документування в проектах розробки програмного забезпечення. Не включає в себе детальну інструкцію з проектування, не містить описів, як будувати діаграми на UML.

Основна мета – ефективне моделювання і документування; але не охоплює програмування та тестування, не включає питання управління проектом, розгортання і супроводу системи. Однак включає в себе перевірку моделі кодом.

2. Agile Unified Process (AUP) спрощена версія IBM Rational Unified Process (RUP), розроблена Скоттом Амблером, яка описує просте і зрозуміле наближення (модель) для створення програмного забезпечення для бізнес-додатків.

3 Agile Data Method – група ітеративних методів розробки програмного забезпечення, в яких вимоги та рішення досягаються в рамках співпраці різних крос-функціональних команд.

4. DSDM заснований на концепції швидкої розробки додатків (Rapid Application Development, RAD). Являє собою ітеративний і інкрементний підхід, який надає особливого значення тривалій участі в процесі користувача/споживача.

5. Essential Unified Process (EssUP).

6. Екстремальне програмування (Extreme programming, XP).

7. Feature driven development (FDD) – функціонально-орієнтована розробка. Використовуване в FDD поняття функції або властивості (feature) Системи досить

близько до поняття прецеденту використання, використовуваному в RUP, істотна відмінність – це додаткове обмеження: «кожна функція повинна допускати реалізацію не більше, ніж за два тижні». Тобто якщо сценарій використання досить малий, його можна вважати функцією. Якщо ж великий, то його треба розбити на декілька відносно незалежних функцій.

8. Getting Real – ітераційний підхід без функціональних специфікацій, що використовується для веб-додатків. У даному методі спершу розробляється інтерфейс програми, а потім її функціональна частина.

9. OpenUP – це ітераційно-інкрементний метод розробки програмного забезпечення. Позиціюється, як легкий і гнучкий варіант RUP. OpenUP ділить життєвий цикл проекту на чотири фази: початкова фаза, фази уточнення, конструювання та передачі. Життєвий цикл проекту забезпечує надання зацікавленим особам та членам колективу точок ознайомлення і прийняття рішень впродовж усього проекту. Це дозволяє ефективно контролювати ситуацію і вчасно приймати рішення про задовільність результатів. План проекту визначає життєвий цикл, а кінцевим результатом є остаточний додаток.

10. Scrum встановлює правила керування процесом розробки та дозволяє використовувати вже існуючі практики кодування, коректуючи вимоги або вносячи тактичні зміни. Використання цієї методології дає можливість виявляти і усувати відхилення від бажаного результату на більш ранніх етапах розробки програмного продукту.

11. Бережлива розробка програмного забезпечення (lean software development). Використовує підходи з концепції бережливого виробництва.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм SHACAL-2, який шифрує дані 256-бітними блоками з використанням 512-бітного ключа. Допускається використання ключів менших розмірів (не менш 128 біт), які доповнюються бітовими нулями до 512 біт.

Шифруємий блок даних ділиться на 8 фрагментів по 32 біта (які позначені буквами *A*...*H*). Алгоритм виконує 64 раунду перетворень, у кожному з яких дані фрагменти обробляються в такий спосіб:

$$T = H_i + S_1(E_i) + Ch(E_i, F_i, G_i) + M_i + K_i,$$

$$H_{i+1} = G_i,$$

$$G_{i+1} = F_i,$$

$$F_{i+1} = E_i,$$

$$E_{i+1} = D_i + T,$$

$$D_{i+1} = C_i,$$

$$C_{i+1} = B_i,$$

$$B_{i+1} = A_i,$$

$$A_{i+1} = T + S_0(A_i) + Maj(A_i, B_i, C_i).$$

де T – тимчасова змінна.

Використовувані функції визначені в такий спосіб:

$$S_0(x) = (x >>> 2) \oplus (x >>> 13) \oplus (x >>> 22),$$

$$S_1(x) = (x >>> 6) \oplus (x >>> 11) \oplus (x >>> 25),$$

$$Ch(x, y, z) = (x \,\&\, y) \oplus (x' \,\&\, z),$$

$$Maj(x, y, z) = (x \,\&\, y) \oplus (x \,\&\, z) \oplus (y \,\&\, z),$$

де >>> – операція побітового циклічного зрушення вправо.

Константи, що модифікують, M_i ($i = 0…63$) наведено нижче (одна за одною від M_0 до M_{63}):

428A2F98	71374491	B5C0FBCF	E9B5DBA5
3956C25B	59F111F1	923F82A4	AB1C5ED5

D807AA98	12835B01	243185BE	550C7DC3
72BE5D74	80DEB1FE	9BDC06A7	C19BF174
E49B69C1	EFBE4786	0FC19DC6	240CA1CC
2DE92C6F	4A7484AA	5CB0A9DC	76F988DA
983E5152	A831C66D	B00327C8	BF597FC7
C6E00BF3	D5A79147	06CA6351	14292967
27B70A85	2E1B2138	4D2C6DFC	53380D13
650A7354	766A0ABB	81C2C92E	92722C85
A2BFE8A1	A81A664B	C24B8B70	C76C51A3
D192E819	D6990624	F40E3585	106AA070
19A4C116	1E376C08	2748774C	34B0BCB5
391C0CB3	4ED8AA4A	5B9CCA4F	682E6FF3
748F82EE	78A5636F	84C87814	8CC70208
90BEFFFA	A4506CEB	BEF9A3F7	C67178F2

Фрагменти розширеного ключа $K_0 \ldots K_{63}$ обчислюються в процесі процедури розширення ключа, що виконується в такий спосіб:

Етап 1. 512-бітний вихідний ключ шифрування ділиться на 16 фрагментів по 32 біта $K_0 \ldots K_{15}$...

Етап 2. Інші фрагменти розширеного ключа $K_{16} \ldots K_{63}$ обчислюються з перших 16 фрагментів у такий спосіб:

$$K_i = O_1(K_{i-2}) + K_{i-7} + O_0(K_{i-15}) + K_{i-16},$$

де функції O_0 і O_1 визначені так:

$$O_0(x) = (x >>> 7) \oplus (x >>> 18) \oplus (x >> 3),$$

$$O_1(x) = (x >>> 17) \oplus (x >>> 19) \oplus (x >> 10),$$

де >> – операція побітового зрушення (не циклічного) вправо.

Раунди розшифрування алгоритму виконуються у зворотній послідовності:

$$T = A_{i+1} + S_0'(B_{i+1}) + Maj'(B_{i+1}, C_{i+1}, D_{i+1}) + 2,$$

$$H_i = T + S_1'(F_{i+1}) + Ch'(F_{i+1}, G_{i+1}, H_{i+1}) + M_i' + K_i' + 4,$$

$$G_i = H_{i+1},$$

$$F_i = G_{i+1},$$

$$E_i = F_{i+1},$$

$$D_i = E_{i+1} + T' + 1,$$

$$C_i = D_{i+1},$$

$$B_i = C_{i+1},$$

$$A_i = B_{i+1}.$$

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ відеоспостереження за технологією VSaaS яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Навігаційне меню: Архівні дані; Налаштування; Довідка.

– Функції представлені у графічному вигляді.

– Розділів обрання відеопотоку.

– Розділу виведення результату роботи системи.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

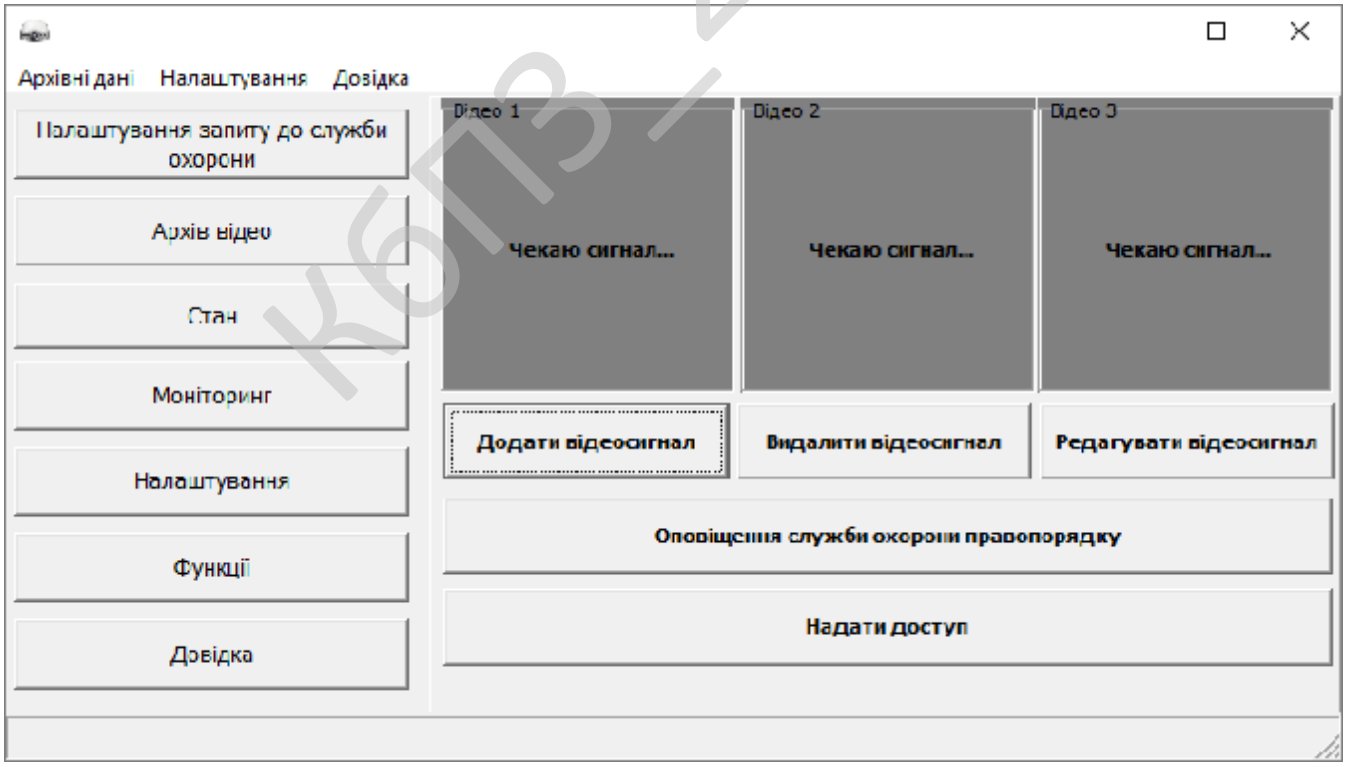

Рисунок 5.1 – Головне вікно ПЗ

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

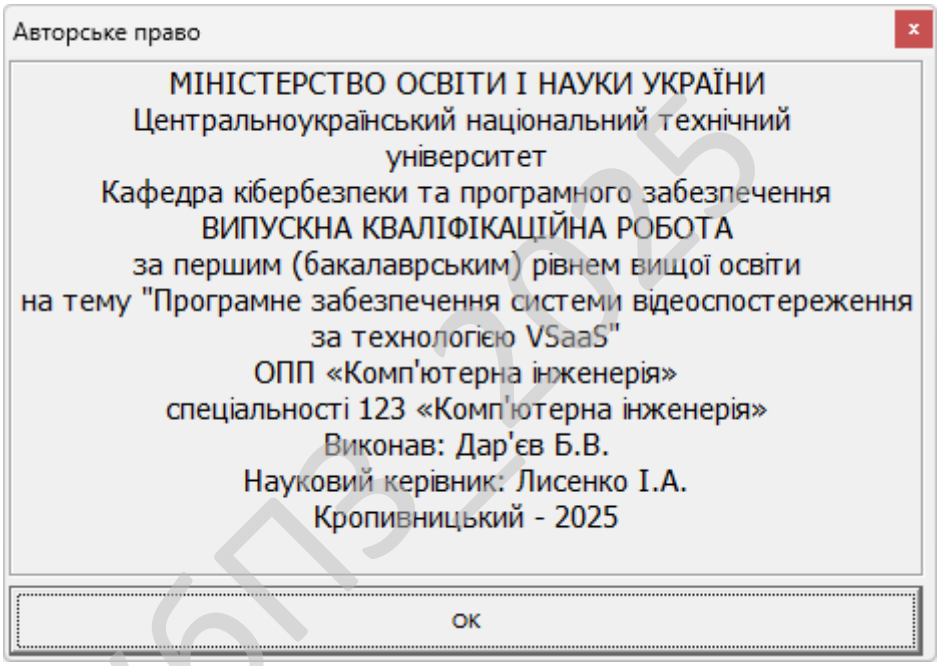

Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Віддалення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати IT рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене IT рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих

процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в IT рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Обрано умови розповсюдження – commercial software.

Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій.

Найважливішою особливістю комерційний програмних продуктів є підтримка великих компаній, прямо зацікавлених у поширенні програм. Багато організацій надають виключно платну підтримку своїх продуктів, такий підхід, як правило, використовують організації надають відкриті вихідні коди. Для продуктів, що розповсюджуються на комерційній основі діють зазвичай безкоштовні служби підтримки, покликані збільшити рівень довіри у клієнтів і потенційних покупців.

Далеко не завжди, але як правило терміни критично важливих змін в комерційних продуктах значно менше, ніж у некомерційних проектів. Це пов'язано з тим, що над комерційним продуктом працюють цілі групи розробників і ця робота є їх основним заняттям. Розробникам-початківцям як правило доводиться шукати додаткові способи заробітку, і це збільшує час, що витрачається на доповнення і зміни програм. Так як основним рушійним фактором створення комерційного ПЗ є одержання прибутку, то комерційні програмні продукти першими заповнюють вільні ніші та пропонують варіанти вирішення завдань відразу по мірі виявлення вакууму в будь-якому секторі ринку.

Окремий вид комерційних програм, коли їх розробка оплачується безпосередньо замовником. Такі програми найчастіше позбавлені всіх переваг комерційних продуктів, оскільки мають обмежений бюджет, але більш адаптовані до вимог замовника, ніж аналоги.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи відеоспостереження за технологією VSaaS.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем відеоспостереження за технологією VSaaS.

– Досліджена система відеоспостереження за технологією VSaaS.

– На основі отриманих результатів досліджень створена програмна реалізація системи відеоспостереження за технологією VSaaS.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання відеоспостереження за технологією VSaaS.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи відеоспостереження за технологією VSaaS. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне

програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм SHACAL-2.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Peter Shirley, Steve Marschner. Fundamentals of Computer Graphics. 2009

2. Михайло Пічугін, Іван Канкін, Володимир Воротніков Комп'ютерна графіка. Навчальний посібник / Центр навчальної літератури 346 с. 2019р.

3. Маценко В.Г. Комп’ютерна графіка: Навчальний посібник. – Чернівці: Рута, 2009 – 343 с.

4. Інженерна комп’ютерна графіка: підручник / В.В. Проців [та ін.] / М-во освіти і науки України, Нац. гірн. унт-т. – Дніпро: НГУ, 2017. – 247 с.

5. Проців В.В. Прикладна комп'ютерна графіка [Текст]: Навч. посібник / В.В. Проців, К.А. Зіборов, К.М. Бас, Г.К. Ванжа; М-во освіти і наук, Нац. гірн. ун-т. - Д.: НГУ, 2016. - 187 с.

6. Kopf, Johannes and Lischinski, Dani. Depixelizing Pixel Art (англ.) // ACM Trans. Graph. – 2011. – Vol. 30, no. 4. – P. 99:1--99:8.

7. Giachetti, Andrea and Asuni, Nicola. Real-Time Artifact-Free Image Upscaling (англ.) // Trans. Img. Proc.. – 2011. – Vol. 20, no. 10. – P. 2760—2768.

8. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

9. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447

10. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

11. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

12. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.

13. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» CEUR Workshop Proceedings, Volume 3187, 2022,

14. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». Sensors (Basel, Switzerland) Volume 22, Issue 16, 6223, 2022.

15. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w

16. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021, Cracow, Poland, 22-25 September 2021. P. 414-418.

17. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». 4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

18. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-

feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

19. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114.

20. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». Journal of theoretical and applied information technology Vol.98. No 21, 2020, P. 3334-3346.

21. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». CEUR Workshop Proceedings Volume 2654, 2020, Pages 1-14.

22. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

23. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

24. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

25. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». CEUR Workshop Proceedings Volume 2616, 2020, Pages 366-379.

26. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

27. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», CEUR Workshop Proceedings Volume 2608, 2020, Pages 646-660.

28. Zhurakovskyi, B., Tsopa, N., Batrak, Y., Odarchenko, R., Smirnova, T «Comparative analysis of modern formats of lossy audio compression». Workshop Proceedings, 2020, 2654, стр. 315-327.

29. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

30. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

31. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

32. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.

33. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

34. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 347-352.

35. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», CEUR Workshop Proceedings Volume 2353, CEUR Workshop Proceedings 2019, Pages 618-629.

36. Smirnov A.A., Kuznetsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», Telecommunications and Radio Engineering. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

37. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». CEUR Workshop Proceedings Volume 2732, 2020, Pages 214-227.

38. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

39. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

40. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

41. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов «Хмарна інформаційна система оцінювання шорсткості з використанням дискретного частотного аналізу макрофотографій». IV міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 15-16 квітня 2021р. – Кропивницький: ЦНТУ. – 2021. – С. 30.

42. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у Кібербезпека та інформаційні технології: монографія. – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

43. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», Кібербезпека: освіта, наука, техніка. № 3(7). С. 43-62. 2020.

44. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 161-172, 2019.

45. О. Смірнов, Є. Деменко, О. Онікійчук, А. Арищенко, Л. Горбачова, «Формування псевдовипадкових послідовностей для приховування даних в зображеннях» Комп'ютерні науки та кібербезпека. № 4. С. 30-37. 2019.

46. Смірнов О.А., Конопліцька-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.

47. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan

D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

48. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

49. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

50. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

51. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". - Випуск 3 (140). - Х.: ХУПС - 2016. - С. 36-39.

52. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп'ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

53. Смірнов О.А., Дрєєв О.М. Порівняння бітових щільностей при використанні різних методів кодування інформації. Збірник наукових праць "Системи обробки інформації". - Випуск 2 (118). т.2. - Х.: ХУПС - 2014. - С. 64-67

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи відеоспостереження за технологією VSaaS.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 46-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи відеоспостереження за технологією VSaaS.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи відеоспостереження за технологією VSaaS;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 71 аркуш.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 5.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Лисенко І.А.

Програмне забезпечення системи відеоспостереження за технологією VSaaS

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 20

Літера: РП

Кропивницький – 2025 року

Основна програма

```
#!/usr/bin/env python3
import time
import threading
import sqlite3
import datetime
import random
from flask import Flask, jsonify, request

# Ініціалізація глобальних змінних для налагодження системи відеоспостереження
DEBUG_MODE = True
# Створення глобального прапорця для припинення роботи потоків
SHUTDOWN_FLAG = False

# Клас, що моделює роботу відеокамери
class Camera:
# Конструктор камери, який приймає унікальний ідентифікатор, місцезнаходження,
роздільну здатність та кількість кадрів за секунду
    def __init__(self, camera_id, location, resolution=(1920, 1080), fps=30):
# Ініціалізація ідентифікатора камери
        self.camera_id = camera_id
# Ініціалізація місцезнаходження камери
        self.location = location
# Ініціалізація роздільної здатності камери
        self.resolution = resolution
# Ініціалізація частоти кадрів
        self.fps = fps
# Прапорець, що вказує, чи активний потік камери
        self.streaming = False
# Зберігання останнього згенерованого кадру
        self.last_frame = None
# Поток для симуляції відеопотоку
        self.thread = None
# Лічильник кадрів для відлагодження
        self.frame_counter = 0

# Метод для запуску відеопотоку камери
    def start_stream(self):
# Перевірка, чи не запущено потік для цієї камери
        if not self.streaming:
# Встановлення прапорця активного потоку
            self.streaming = True
# Створення нового потоку для симуляції відео
            self.thread = threading.Thread(target=self._simulate_stream)
# Запуск потоку симуляції
            self.thread.start()
# Вивід повідомлення про запуск відеопотоку, якщо увімкнено режим налагодження
            if DEBUG_MODE:
                print("Запущено потік камери:", self.camera_id)

# Метод для зупинки відеопотоку камери
    def stop_stream(self):
# Перевірка, чи активний потік перед зупинкою
        if self.streaming:
# Зупинка потоку
            self.streaming = False
# Очікування завершення потоку
            if self.thread is not None:
                self.thread.join()
# Вивід повідомлення про зупинку потоку, якщо увімкнено режим налагодження
                if DEBUG_MODE:
                    print("Зупинено потік камери:", self.camera_id)
```

```
# Приватний метод, що симулює роботу відеопотоку камери
    def _simulate_stream(self):
# Основний цикл симуляції потоку
        while self.streaming and not SHUTDOWN_FLAG:
# Генерація нового кадру
            self.last_frame = self._generate_frame()
# Збільшення лічильника кадрів
            self.frame_counter += 1
# Обробка отриманого кадру для виявлення подій
            self._process_frame(self.last_frame)
# Затримка для імітації кількості кадрів за секунду
            time.sleep(1 / self.fps)
# Вивід відлагоджувальної інформації про кадри, якщо увімкнено режим
налагодження
        if DEBUG_MODE:
            print("Вихід із потоку камери:", self.camera_id)

# Приватний метод для генерації симульованого кадру відео
    def _generate_frame(self):
# Генерація випадкового значення для симуляції кадру
        frame = random.randint(0, 255)
# Повернення згенерованого кадру
        return frame

# Приватний метод для обробки кадру та виявлення подій
    def _process_frame(self, frame):
# Симуляція аналізу кадру для виявлення руху
        if random.random() < 0.01:
# Якщо виявлено подію, відправка повідомлення до системи відеоспостереження
            SurveillanceSystem.get_instance().event_logger.log_event(
                camera_id=self.camera_id,
                event_type="Motion Detected",
                event_details="Виявлено рух в зоні " + self.location
            )
# Вивід повідомлення про виявлену подію, якщо увімкнено режим налагодження
            if DEBUG_MODE:
                print("Виявлено рух на камері:", self.camera_id)

# Додатковий метод для отримання стану камери
    def get_status(self):
# Повернення словника зі станом камери
        return {
            "camera_id": self.camera_id,
            "location": self.location,
            "resolution": self.resolution,
            "fps": self.fps,
            "streaming": self.streaming,
            "frame_counter": self.frame_counter
        }

# Клас для управління декількома камерами в системі відеоспостереження
class CameraManager:
# Конструктор класу, який ініціалізує список камер
    def __init__(self):
# Словник для зберігання об'єктів камер за їх ідентифікаторами
        self.cameras = {}
# Лічильник для відлагодження кількості доданих камер
        self.total_cameras = 0

# Метод для додавання нової камери до системи
    def add_camera(self, camera):
# Додавання камери до словника за її унікальним ідентифікатором
```

```
        self.cameras[camera.camera_id] = camera
# Збільшення загальної кількості камер
        self.total_cameras += 1
# Вивід інформації про додану камеру, якщо увімкнено режим налагодження
        if DEBUG_MODE:
            print("Додано камеру:", camera.camera_id)

# Метод для видалення камери зі системи за її ідентифікатором
    def remove_camera(self, camera_id):
# Перевірка наявності камери у системі
        if camera_id in self.cameras:
# Зупинка відеопотоку перед видаленням камери
            self.cameras[camera_id].stop_stream()
# Видалення камери зі словника
            del self.cameras[camera_id]
# Вивід інформації про видалення камери, якщо увімкнено режим налагодження
            if DEBUG_MODE:
                print("Видалено камеру:", camera_id)

# Метод для запуску потоків усіх камер
    def start_all_streams(self):
# Ітерація по всіх камерах у системі
        for camera in self.cameras.values():
# Запуск потоку для кожної камери
            camera.start_stream()
# Вивід відлагоджувального повідомлення про запуск всіх потоків
        if DEBUG_MODE:
            print("Запущено всі відеопотоки камер.")

# Метод для зупинки потоків усіх камер
    def stop_all_streams(self):
# Ітерація по всіх камерах для зупинки потоків
        for camera in self.cameras.values():
# Зупинка потоку для кожної камери
            camera.stop_stream()
# Вивід відлагоджувального повідомлення про зупинку всіх потоків
        if DEBUG_MODE:
            print("Зупинено всі відеопотоки камер.")

# Метод для отримання статусу конкретної камери
    def query_camera_status(self, camera_id):
# Перевірка, чи існує камера з заданим ідентифікатором
        if camera_id in self.cameras:
# Отримання об'єкта камери
            camera = self.cameras[camera_id]
# Повернення інформації про камеру
            return camera.get_status()
# Повернення None, якщо камера не знайдена
        return None

# Метод для отримання списку всіх камер у системі
    def list_cameras(self):
# Ініціалізація списку для зберігання інформації про камери
        camera_list = []
# Ітерація по всіх камерах у системі
        for camera in self.cameras.values():
# Додавання статусу кожної камери до списку
            camera_list.append(camera.get_status())
# Повернення списку камер
        return camera_list

# Клас для логування подій у базі даних SQLite
class EventLogger:
```

```
# Конструктор класу для ініціалізації бази даних
    def __init__(self, db_path="events.db"):
# Збереження шляху до бази даних
        self.db_path = db_path
# Створення підключення до бази даних з можливістю роботи в багатопоточному
режимі
        self.connection = sqlite3.connect(self.db_path, check_same_thread=False)
# Виклик методу створення таблиці подій
        self._create_table()
# Ініціалізація об’єкта блокування для синхронізації доступу до бази даних
        self.lock = threading.Lock()

# Приватний метод для створення таблиці подій, якщо вона не існує
    def _create_table(self):
# Отримання курсора для виконання SQL-запитів
        cursor = self.connection.cursor()
# Виконання запиту на створення таблиці подій
        cursor.execute("""
            CREATE TABLE IF NOT EXISTS events (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                timestamp TEXT,
                camera_id TEXT,
                event_type TEXT,
                event_details TEXT
            )
        """)
# Фіксація змін у базі даних
        self.connection.commit()
# Вивід інформації про створення таблиці, якщо увімкнено режим налагодження
        if DEBUG_MODE:
            print("Таблиця подій створена або вже існує.")

# Метод для логування події у базі даних
    def log_event(self, camera_id, event_type, event_details):
# Використання блокування для забезпечення потокобезпеки при доступі до бази
даних
        with self.lock:
# Отримання курсора для виконання SQL-запиту
            cursor = self.connection.cursor()
# Отримання поточного часу у форматі ISO
            timestamp = datetime.datetime.now().isoformat()
# Виконання SQL-запиту для вставки нової події у таблицю
            cursor.execute("INSERT INTO events (timestamp, camera_id,
event_type, event_details) VALUES (?, ?, ?, ?)",
                           (timestamp, camera_id, event_type, event_details))
# Фіксація змін у базі даних
            self.connection.commit()
# Вивід інформації про зареєстровану подію, якщо увімкнено режим налагодження
            if DEBUG_MODE:
                print("Зареєстровано подію для камери:", camera_id)

# Метод для отримання списку подій із бази даних
    def query_events(self, camera_id=None):
# Використання блокування для безпечного доступу до бази даних
        with self.lock:
# Отримання курсора для виконання SQL-запиту
            cursor = self.connection.cursor()
# Перевірка, чи заданий фільтр за ідентифікатором камери
            if camera_id:
# Виконання SQL-запиту для отримання подій конкретної камери
                cursor.execute("SELECT * FROM events WHERE camera_id=?",
(camera_id,))
            else:
```

```
# Виконання SQL-запиту для отримання усіх подій
                cursor.execute("SELECT * FROM events")
# Отримання всіх рядків з результату запиту
            rows = cursor.fetchall()
# Ініціалізація списку для зберігання даних про події
            events = []
# Обробка кожного рядка з отриманих даних
            for row in rows:
# Додавання інформації про подію до списку
                events.append({
                    "id": row[0],
                    "timestamp": row[1],
                    "camera_id": row[2],
                    "event_type": row[3],
                    "event_details": row[4]
                })
# Повернення списку подій
            return events

# Клас, що представляє головну систему відеоспостереження (патерн Singleton)
class SurveillanceSystem:
# Приватна змінна для зберігання єдиного екземпляру класу
    _instance = None

# Метод для отримання єдиного екземпляру системи
    @classmethod
    def get_instance(cls):
# Перевірка, чи вже існує екземпляр системи
        if cls._instance is None:
# Створення нового екземпляру, якщо він відсутній
            cls._instance = SurveillanceSystem()
# Повернення єдиного екземпляру системи
        return cls._instance

# Конструктор системи відеоспостереження
    def __init__(self):
# Захист від створення декількох екземплярів системи
        if SurveillanceSystem._instance is not None:
            raise Exception("Цей клас є синглтоном і вже існує екземпляр!")
# Ініціалізація менеджера камер
        self.camera_manager = CameraManager()
# Ініціалізація логера подій
        self.event_logger = EventLogger()
# Ініціалізація веб-сервера API за допомогою Flask
        self.api_app = Flask(__name__)
# Налаштування маршрутів для API
        self._setup_api_routes()
# Прапорець, що вказує на активність системи
        self.running = False
# Ініціалізація списку для зберігання додаткових задач
        self.extra_tasks = []
# Вивід повідомлення про ініціалізацію системи, якщо увімкнено режим
налагодження
        if DEBUG_MODE:
            print("Ініціалізовано систему відеоспостереження.")

# Приватний метод для налаштування маршрутів веб-сервера API
    def _setup_api_routes(self):
# Маршрут для отримання списку всіх камер
        @self.api_app.route('/api/cameras', methods=['GET'])
        def api_list_cameras():
# Отримання списку камер із менеджера камер
            cameras = self.camera_manager.list_cameras()
```

```
# Повернення даних у форматі JSON
            return jsonify(cameras)
# Маршрут для отримання стану конкретної камери за її ідентифікатором
        @self.api_app.route('/api/camera/<camera_id>', methods=['GET'])
        def api_camera_status(camera_id):
# Отримання статусу камери
            status = self.camera_manager.query_camera_status(camera_id)
# Перевірка наявності камери та повернення відповідного результату
            if status:
                return jsonify(status)
            return jsonify({"error": "Камеру не знайдено"}), 404
# Маршрут для отримання списку зареєстрованих подій
        @self.api_app.route('/api/events', methods=['GET'])
        def api_get_events():
# Отримання ідентифікатора камери з параметрів запиту
            camera_id = request.args.get('camera_id')
# Отримання подій за допомогою логера подій
            events = self.event_logger.query_events(camera_id)
# Повернення подій у форматі JSON
            return jsonify(events)
# Маршрут для додавання нової камери до системи
        @self.api_app.route('/api/camera', methods=['POST'])
        def api_add_camera():
# Отримання даних з тіла запиту у форматі JSON
            data = request.json
# Отримання ідентифікатора камери з отриманих даних
            camera_id = data.get('camera_id')
# Отримання місцезнаходження камери з отриманих даних
            location = data.get('location')
# Отримання роздільної здатності, якщо вона задана, або використання значення за
замовчуванням
            resolution = data.get('resolution', (1920, 1080))
# Отримання значення fps, якщо воно задане, або використання значення за
замовчуванням
            fps = data.get('fps', 30)
# Перевірка наявності обов’язкових даних для створення камери
            if camera_id and location:
# Створення об’єкта камери
                camera = Camera(camera_id, location, resolution, fps)
# Додавання камери до менеджера камер
                self.camera_manager.add_camera(camera)
# Запуск відеопотоку для новододаної камери
                camera.start_stream()
# Повернення повідомлення про успішне додавання камери
                return jsonify({"status": "Камеру додано"}), 201
# Повернення повідомлення про помилку у випадку відсутності даних
            return jsonify({"error": "Невірні дані"}), 400
# Маршрут для видалення камери з системи за її ідентифікатором
        @self.api_app.route('/api/camera/<camera_id>', methods=['DELETE'])
        def api_remove_camera(camera_id):
# Видалення камери з менеджера камер
            self.camera_manager.remove_camera(camera_id)
# Повернення повідомлення про успішне видалення камери
            return jsonify({"status": "Камеру видалено"})

# Метод для запуску системи відеоспостереження
    def start_system(self):
# Встановлення прапорця активності системи
        self.running = True
# Запуск усіх відеопотоків камер
        self.camera_manager.start_all_streams()
# Створення окремого потоку для запуску веб-сервера API
        api_thread = threading.Thread(target=self._run_api_server)
```

```
# Встановлення демона для потоку API
        api_thread.daemon = True
# Запуск потоку веб-сервера
        api_thread.start()
# Додавання потоку веб-сервера до списку додаткових задач
        self.extra_tasks.append(api_thread)
# Вивід повідомлення про запуск системи, якщо увімкнено режим налагодження
        if DEBUG_MODE:
            print("Система відеоспостереження запущена.")

# Метод для зупинки системи відеоспостереження
    def stop_system(self):
# Встановлення прапорця припинення роботи системи
        self.running = False
# Зупинка усіх відеопотоків камер
        self.camera_manager.stop_all_streams()
# Вивід повідомлення про зупинку системи, якщо увімкнено режим налагодження
        if DEBUG_MODE:
            print("Система відеоспостереження зупинена.")

# Приватний метод для запуску веб-сервера API
    def _run_api_server(self):
# Запуск веб-сервера API на заданому хості та порту
        self.api_app.run(host='0.0.0.0', port=5000)

# Функція для симуляції періодичних запитів до системи для отримання статистики
def simulate_queries():
# Отримання єдиного екземпляру системи відеоспостереження
    system = SurveillanceSystem.get_instance()
# Безкінечний цикл для періодичних запитів
    while not SHUTDOWN_FLAG:
# Отримання списку камер із менеджера камер
        cameras = system.camera_manager.list_cameras()
# Вивід статусу кожної камери
        for cam in cameras:
            print("Статус камери:", cam['camera_id'], "-> Стримінг:",
                  "Активний" if cam['streaming'] else "Зупинено", "-> Лічильник
кадрів:", cam['frame_counter'])
# Отримання списку подій із логера подій
        events = system.event_logger.query_events()
# Вивід інформації про загальну кількість подій
        print("Загальна кількість подій:", len(events))
# Затримка перед наступним запитом
        time.sleep(10)
# Вивід повідомлення про завершення циклу запитів
    if DEBUG_MODE:
        print("Завершено цикл симуляції запитів.")

# Функція для ініціалізації тестових камер у системі
def initialize_sample_cameras():
# Отримання єдиного екземпляру системи відеоспостереження
    system = SurveillanceSystem.get_instance()
# Створення першої тестової камери з унікальним ідентифікатором
    cam1 = Camera("CAM001", "Вхідні ворота", (1920, 1080), 25)
# Додавання першої камери до системи
    system.camera_manager.add_camera(cam1)
# Створення другої тестової камери
    cam2 = Camera("CAM002", "Лобі", (1280, 720), 20)
# Додавання другої камери до системи
    system.camera_manager.add_camera(cam2)
# Створення третьої тестової камери
    cam3 = Camera("CAM003", "Паркінг", (1920, 1080), 30)
# Додавання третьої камери до системи
```

```
    system.camera_manager.add_camera(cam3)
# Запуск відеопотоків для всіх тестових камер
    system.camera_manager.start_all_streams()
# Вивід повідомлення про завершення ініціалізації тестових камер
    if DEBUG_MODE:
        print("Ініціалізація тестових камер завершена.")

# Функція для симуляції періодичних завдань з технічного обслуговування системи
def scheduled_maintenance():
# Отримання єдиного екземпляру системи відеоспостереження
    system = SurveillanceSystem.get_instance()
# Безкінечний цикл для виконання завдань обслуговування
    while not SHUTDOWN_FLAG:
# Вивід повідомлення про початок завдання технічного обслуговування
        print("Виконується заплановане обслуговування системи...")
# Отримання поточного списку подій із логера подій
        events = system.event_logger.query_events()
# Вивід інформації про кількість подій у базі даних
        print("Перевірка обслуговування: Кількість подій у базі даних:",
len(events))
# Імітація процесу очищення бази даних або архівування подій
        time.sleep(60)
# Вивід повідомлення про завершення циклу технічного обслуговування
    if DEBUG_MODE:
        print("Завершено цикл технічного обслуговування.")

# Додаткова функція для симуляції відправлення сповіщень у разі виявлення подій
def send_alert(camera_id, event_type, event_details):
# Формування повідомлення сповіщення
    alert_message = f"Сповіщення! Камера {camera_id} зафіксувала подію:
{event_type}. Деталі: {event_details}"
# Вивід сповіщення у консоль (імітація відправлення email або SMS)
    print(alert_message)
# Логування сповіщення у режимі відлагодження
    if DEBUG_MODE:
        print("Сповіщення відправлено для камери:", camera_id)

# Додаткова функція для періодичної перевірки стану системи
def system_health_check():
# Отримання єдиного екземпляру системи відеоспостереження
    system = SurveillanceSystem.get_instance()
# Безкінечний цикл для перевірки стану системи
    while not SHUTDOWN_FLAG:
# Перевірка загального стану системи
        cameras = system.camera_manager.list_cameras()
# Вивід загальної кількості камер у системі
        print("Стан системи: загальна кількість камер =", len(cameras))
# Затримка перед наступною перевіркою
        time.sleep(30)
# Вивід повідомлення про завершення перевірки стану
    if DEBUG_MODE:
        print("Цикл перевірки стану системи завершено.")

# Додаткова функція для симуляції виконання запитів до бази даних із складними
умовами
def advanced_event_query(camera_id=None, start_time=None, end_time=None):
# Отримання єдиного екземпляру системи відеоспостереження
    system = SurveillanceSystem.get_instance()
# Отримання курсора бази даних для виконання розширеного запиту
    with system.event_logger.lock:
        cursor = system.event_logger.connection.cursor()
# Формування базового SQL-запиту
        base_query = "SELECT * FROM events WHERE 1=1"
```

```
# Список параметрів для SQL-запиту
        params = []
# Додавання умови за ідентифікатором камери, якщо задано
        if camera_id:
            base_query += " AND camera_id = ?"
            params.append(camera_id)
# Додавання умови за початковим часом, якщо задано
        if start_time:
            base_query += " AND timestamp >= ?"
            params.append(start_time)
# Додавання умови за кінцевим часом, якщо задано
        if end_time:
            base_query += " AND timestamp <= ?"
            params.append(end_time)
# Виконання складного SQL-запиту
        cursor.execute(base_query, tuple(params))
# Отримання всіх рядків результату запиту
        rows = cursor.fetchall()
# Формування списку результатів
        result_events = []
        for row in rows:
            result_events.append({
                "id": row[0],
                "timestamp": row[1],
                "camera_id": row[2],
                "event_type": row[3],
                "event_details": row[4]
            })
# Повернення списку знайдених подій
        return result_events

# Головна функція для запуску програми системи відеоспостереження
def main():
# Вивід стартового повідомлення системи, якщо увімкнено режим налагодження
    if DEBUG_MODE:
        print("Старт системи відеоспостереження...")
# Ініціалізація тестових камер у системі
    initialize_sample_cameras()
# Отримання єдиного екземпляру системи відеоспостереження
    system = SurveillanceSystem.get_instance()
# Запуск системи, що включає API-сервер та відеопотоки
    system.start_system()
# Створення окремого потоку для симуляції періодичних запитів до системи
    query_thread = threading.Thread(target=simulate_queries)
# Встановлення демона для потоку запитів
    query_thread.daemon = True
# Запуск потоку запитів
    query_thread.start()
# Додавання потоку запитів до списку додаткових задач
    system.extra_tasks.append(query_thread)
# Створення окремого потоку для виконання запланованого технічного
обслуговування
    maintenance_thread = threading.Thread(target=scheduled_maintenance)
# Встановлення демона для потоку технічного обслуговування
    maintenance_thread.daemon = True
# Запуск потоку технічного обслуговування
    maintenance_thread.start()
# Додавання потоку технічного обслуговування до списку додаткових задач
    system.extra_tasks.append(maintenance_thread)
# Створення окремого потоку для перевірки стану системи
    health_thread = threading.Thread(target=system_health_check)
# Встановлення демона для потоку перевірки стану
    health_thread.daemon = True
```

```
# Запуск потоку перевірки стану системи
    health_thread.start()
# Додавання потоку перевірки стану до списку додаткових задач
    system.extra_tasks.append(health_thread)
# Головний цикл для утримання роботи програми
    try:
        while True:
# Перевірка основного циклу з перериванням на вимкнення
            time.sleep(1)
    except KeyboardInterrupt:
# Встановлення прапорця завершення роботи системи
        global SHUTDOWN_FLAG
        SHUTDOWN_FLAG = True
# Зупинка системи відеоспостереження
        system.stop_system()
# Вивід повідомлення про завершення роботи програми
        print("Ініційовано завершення роботи системи відеоспостереження.")
# Додаткове очікування для коректного завершення потоків
        time.sleep(3)
# Вивід фінального повідомлення про успішне завершення роботи
    if DEBUG_MODE:
        print("Програма завершила роботу.")

# Виклик головної функції при запуску скрипту
if __name__ == "__main__":
    main()
```

Файл vas.py

```
from ultralytics import YOLO
import cv2 as cv
from PIL import Image
import numpy as np
import torch
import os

class VAS:
    def __init__(self, model_name, video_path):
        self.device = 'cuda' if torch.cuda.is_available() else 'cpu'
        self.model_name = model_name
        self.video_path = video_path #"Example video.mp4"
        self.model = self.load_model()
        self.font = cv.FONT_HERSHEY_SIMPLEX
        self.location = (0,50)
        self.distance = (0,10)
        self.fontScale = 1
        self.fontColor = (0,0,255)
        self.lineType = 3

    def load_model(self):
        model = YOLO(self.model_name) #'yolov8m.pt'
        model.to(self.device)

        return model

    def predict(self, frame):
        return self.model(frame, verbose = False)

    def plot_boxes(self, results, annotated_frame):
        for r in results:
            result = r.boxes.cpu()
            key_list = r.names
            unique, counts = np.unique(result.cls, return_counts=True)
            number = dict(zip(unique,counts))
            location = (0,70)
            cv.putText(annotated_frame,'There are:',
                               (0,50),
                               self.font,
                               self.fontScale,
                               self.fontColor,
                               self.lineType)
            for i in unique:
                location = (location[0],location[1]+35)
                cv.putText(annotated_frame,'{} {}(s)'.format(number[i],
key_list[int(i)]),
                               location,
                               self.font,
                               self.fontScale,
                               self.fontColor,
                               self.lineType)

        return annotated_frame

    def __call__(self):
        cap = cv.VideoCapture(self.video_path)
        frame_width = int(cap.get(3))
        frame_height = int(cap.get(4))
        fps = int(cap.get(5))

        output_folder = './static/annotatedVideo'
```

```
        if not os.path.exists(output_folder):
            os.makedirs(output_folder)
        output_path = os.path.join(output_folder, 'output.mp4')

        # Write frame
        size = (frame_width, frame_height)

        output = cv.VideoWriter(output_path,
                                cv.VideoWriter_fourcc(*'H264'),
                                fps, size)

        # Loop through the video frames
        while cap.isOpened():
            # Read a frame from the video
            success, frame = cap.read()
            if not success:
                break

            # Run YOLOv8 inference on the frame
            results = self.predict(frame)

            # Visualize the results on the frame
            annotated_frame = results[0].plot()

            count_annotated_frame = self.plot_boxes(results,annotated_frame)
            # cv.imshow('YOLO V8 Detection', count_annotated_frame)
            # if cv.waitKey(1) & 0xFF == ord(' '):
            #     break
            output.write(count_annotated_frame)

        output.release()
        cap.release()
        cv.destroyAllWindows()

# analyzed_video = VAS('yolov8n.pt', 'Examples/ExampleVideo.mp4')
# analyzed_video()
```

Файл FaceRecognitionModule.py

```
import cv2
import face_recognition
import numpy as np
import sqlite3
import datetime
import threading
import time
import random
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression

class FaceRecognitionModule:
    def __init__(self):
        self.known_face_encodings = []
        self.known_face_names = []
    def load_known_faces(self, image_files, names):
        for image_file, name in zip(image_files, names):
            img = face_recognition.load_image_file(image_file)
            encs = face_recognition.face_encodings(img)
            if len(encs) > 0:
                self.known_face_encodings.append(encs[0])
                self.known_face_names.append(name)
    def recognize_faces(self, frame):
        rgb_frame = frame[:, :, ::-1]
        locations = face_recognition.face_locations(rgb_frame)
        encodings = face_recognition.face_encodings(rgb_frame, locations)
        results = []
        for (top, right, bottom, left), encoding in zip(locations, encodings):
            matches = face_recognition.compare_faces(self.known_face_encodings,
encoding)
            distances =
face_recognition.face_distance(self.known_face_encodings, encoding)
            name = "Unknown"
            if len(distances) > 0:
                best_index = np.argmin(distances)
                if matches[best_index]:
                    name = self.known_face_names[best_index]
            results.append({"location": (top, right, bottom, left), "name":
name})
        return results
    def process_video_stream(self, source=0):
        cap = cv2.VideoCapture(source)
        while True:
            ret, frame = cap.read()
            if not ret:
                break
            faces = self.recognize_faces(frame)
            for face in faces:
                top, right, bottom, left = face["location"]
                cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255),
2)
                cv2.putText(frame, face["name"], (left, top-10),
cv2.FONT_HERSHEY_DUPLEX, 0.5, (255, 255, 255), 1)
            cv2.imshow("Face Recognition", frame)
            if cv2.waitKey(1) & 0xFF == ord("q"):
                break
        cap.release()
        cv2.destroyAllWindows()

class BehaviorAnalyticsModule:
```

```
    def __init__(self):
        self.tracks = []
    def add_track(self, obj_id, pos, ts):
        self.tracks.append({"id": obj_id, "position": pos, "timestamp": ts})
    def get_all_tracks(self):
        return self.tracks
    def cluster_tracks(self):
        pos_list = [track["position"] for track in self.tracks]
        if not pos_list:
            return {}
        X = np.array(pos_list)
        X_scaled = StandardScaler().fit_transform(X)
        clustering = DBSCAN(eps=0.5, min_samples=2).fit(X_scaled)
        labels = clustering.labels_
        clusters = {}
        for i, label in enumerate(labels):
            if label not in clusters:
                clusters[label] = []
            clusters[label].append(self.tracks[i])
        return clusters
    def analyze_anomalies(self):
        clusters = self.cluster_tracks()
        anomalies = []
        for label, group in clusters.items():
            if label == -1 or len(group) < 3:
                anomalies.extend(group)
        return anomalies
    def simulate_tracks(self, num_objs=10, num_points=20):
        self.tracks = []
        for i in range(num_objs):
            base_x = random.uniform(0, 100)
            base_y = random.uniform(0, 100)
            for j in range(num_points):
                delta_x = random.uniform(-5, 5)
                delta_y = random.uniform(-5, 5)
                ts = datetime.datetime.now() + datetime.timedelta(seconds=j)
                self.add_track(i, (base_x + delta_x, base_y + delta_y), ts)
                time.sleep(0.005)
    def run_behavior_analysis(self):
        self.simulate_tracks()
        return self.analyze_anomalies()

class IncidentPredictionModule:
    def __init__(self):
        self.data = []
        self.model = None
    def add_data(self, features, incident):
        self.data.append({"features": features, "incident": incident})
    def prepare_data(self):
        X, y = [], []
        for d in self.data:
            X.append(d["features"])
            y.append(1 if d["incident"] else 0)
        return np.array(X), np.array(y)
    def train(self):
        X, y = self.prepare_data()
        if len(X) == 0:
            return
        self.model = LinearRegression()
        self.model.fit(X, y)
    def predict(self, features):
        if self.model is None:
            return 0
```

```
        pred = self.model.predict([features])
        return pred[0]
    def simulate_data(self, samples=100):
        self.data = []
        for _ in range(samples):
            f1 = random.uniform(0, 10)
            f2 = random.uniform(0, 10)
            incident = (f1 + f2) > 10
            self.add_data([f1, f2], incident)
    def run_prediction(self, features):
        self.simulate_data()
        self.train()
        return self.predict(features)

class AdvancedEventLogger:
    def __init__(self, db="events.db"):
        self.conn = sqlite3.connect(db)
        self.initialize_db()
    def initialize_db(self):
        cur = self.conn.cursor()
        cur.execute("CREATE TABLE IF NOT EXISTS events (id INTEGER PRIMARY KEY,
timestamp TEXT, event_type TEXT, details TEXT)")
        self.conn.commit()
    def log(self, event_type, details):
        cur = self.conn.cursor()
        ts = datetime.datetime.now().isoformat()
        cur.execute("INSERT INTO events (timestamp, event_type, details) VALUES
(?, ?, ?)", (ts, event_type, details))
        self.conn.commit()
    def fetch_events(self):
        cur = self.conn.cursor()
        cur.execute("SELECT id, timestamp, event_type, details FROM events")
        rows = cur.fetchall()
        events = []
        for row in rows:
            events.append({"id": row[0], "timestamp": row[1], "event_type":
row[2], "details": row[3]})
        return events
    def clear_events(self):
        cur = self.conn.cursor()
        cur.execute("DELETE FROM events")
        self.conn.commit()
    def backup(self, backup_db="backup_events.db"):
        bck = sqlite3.connect(backup_db)
        with bck:
            self.conn.backup(bck)
        bck.close()
    def run_demo_logging(self):
        for i in range(20):
            self.log("Type" + str(i), "Detail " + str(i))
            time.sleep(0.01)
        return self.fetch_events()

class RemoteHardwareMonitor:
    def __init__(self):
        self.sensors = {}
        self.active = False
    def add_sensor(self, sensor_id, sensor_type, read_func):
        self.sensors[sensor_id] = {"type": sensor_type, "read": read_func,
"value": None}
    def poll_sensor(self, sensor_id):
        if sensor_id in self.sensors:
            val = self.sensors[sensor_id]["read"]()
```

```
                self.sensors[sensor_id]["value"] = val
                return val
        return None
    def poll_all(self):
        for sid in self.sensors:
            self.poll_sensor(sid)
    def start_monitoring(self, interval=1):
        self.active = True
        while self.active:
            self.poll_all()
            time.sleep(interval)
    def stop_monitoring(self):
        self.active = False
    def get_status(self):
        status = {}
        for sid, info in self.sensors.items():
            status[sid] = info["value"]
        return status
    def simulate_reading(self):
        return random.uniform(0, 100)
    def run_monitoring_thread(self, interval=1):
        thread = threading.Thread(target=self.start_monitoring,
args=(interval,))
        thread.start()
        return thread

def main():
    fr = FaceRecognitionModule()
    ba = BehaviorAnalyticsModule()
    ip = IncidentPredictionModule()
    el = AdvancedEventLogger()
    rhm = RemoteHardwareMonitor()
    rhm.add_sensor("temp_sensor", "temperature", rhm.simulate_reading)
    rhm.add_sensor("hum_sensor", "humidity", rhm.simulate_reading)
    monitor_thread = rhm.run_monitoring_thread(0.5)
    incident_score = ip.run_prediction([5, 7])
    anomalies = ba.run_behavior_analysis()
    logged_events = el.run_demo_logging()
    sensor_status = rhm.get_status()
    print("Incident Score:", incident_score)
    print("Anomalies:", anomalies)
    print("Events:", logged_events)
    print("Sensors:", sensor_status)
    rhm.stop_monitoring()
    monitor_thread.join()

if __name__ == "__main__":
    main()
```

Файл CloudStorageModule.py

```
import numpy as np
import random
import time
import threading
import sqlite3
import datetime
from sklearn.cluster import KMeans
from sklearn.ensemble import RandomForestClassifier

class CloudStorageModule:
    def __init__(self):
        self.storage = {}
    def upload_file(self, file_name, data):
        self.storage[file_name] = {"data": data, "timestamp":
datetime.datetime.now().isoformat()}
        return True
    def download_file(self, file_name):
        return self.storage.get(file_name, None)
    def list_files(self):
        return list(self.storage.keys())
    def delete_file(self, file_name):
        if file_name in self.storage:
            del self.storage[file_name]
            return True
        return False
    def simulate_large_upload(self, num_files):
        for i in range(num_files):
            fname = f"file_{i}.dat"
            data = np.random.bytes(1024)
            self.upload_file(fname, data)
            time.sleep(0.01)
        return self.list_files()

class AccessControlModule:
    def __init__(self):
        self.users = {}
        self.roles = {}
    def add_user(self, username, password, role):
        self.users[username] = {"password": password, "role": role}
        if role not in self.roles:
            self.roles[role] = []
        self.roles[role].append(username)
    def authenticate(self, username, password):
        user = self.users.get(username)
        if user and user["password"] == password:
            return True
        return False
    def change_password(self, username, new_password):
        if username in self.users:
            self.users[username]["password"] = new_password
            return True
        return False
    def check_permission(self, username, required_role):
        user = self.users.get(username)
        if user and user["role"] == required_role:
            return True
        return False
    def list_users_by_role(self, role):
        return self.roles.get(role, [])
    def simulate_access_control(self):
```

```
        self.add_user("alice", "pass1", "admin")
        self.add_user("bob", "pass2", "user")
        self.add_user("charlie", "pass3", "user")
        auth1 = self.authenticate("alice", "pass1")
        auth2 = self.authenticate("bob", "wrong")
        perm1 = self.check_permission("alice", "admin")
        perm2 = self.check_permission("charlie", "admin")
        return auth1, auth2, perm1, perm2

class AudioAnalysisModule:
    def __init__(self):
        self.audio_data = None
        self.sample_rate = 44100
    def load_audio(self, duration_sec):
        t = np.linspace(0, duration_sec, int(self.sample_rate * duration_sec),
endpoint=False)
        self.audio_data = 0.5 * np.sin(2 * np.pi * 440 * t) + 0.3 * np.sin(2 *
np.pi * 880 * t)
        return self.audio_data
    def compute_fft(self):
        if self.audio_data is None:
            return None
        fft_result = np.fft.fft(self.audio_data)
        freqs = np.fft.fftfreq(len(fft_result), 1 / self.sample_rate)
        return freqs, np.abs(fft_result)
    def detect_peaks(self, fft_magnitude, threshold=50):
        peaks = []
        for i, val in enumerate(fft_magnitude):
            if val > threshold:
                peaks.append(i)
        return peaks
    def analyze_audio(self, duration_sec):
        self.load_audio(duration_sec)
        freqs, fft_magnitude = self.compute_fft()
        peaks = self.detect_peaks(fft_magnitude)
        return freqs[peaks], fft_magnitude[peaks]

class VideoQualityController:
    def __init__(self):
        self.network_speed = 100.0
        self.resolution = (1920, 1080)
        self.fps = 30
    def update_network_speed(self, speed):
        self.network_speed = speed
    def adjust_quality(self):
        if self.network_speed < 20:
            self.resolution = (640, 360)
            self.fps = 15
        elif self.network_speed < 50:
            self.resolution = (1280, 720)
            self.fps = 20
        elif self.network_speed < 80:
            self.resolution = (1600, 900)
            self.fps = 25
        else:
            self.resolution = (1920, 1080)
            self.fps = 30
    def simulate_network_conditions(self, iterations=10):
        qualities = []
        for _ in range(iterations):
            speed = random.uniform(10, 100)
            self.update_network_speed(speed)
            self.adjust_quality()
```

```
            qualities.append((speed, self.resolution, self.fps))
            time.sleep(0.05)
        return qualities

class ColorRecognitionModule:
    def __init__(self):
        self.k = 3
    def extract_dominant_colors(self, image):
        pixels = image.reshape((-1, 3))
        pixels = np.float32(pixels)
        kmeans = KMeans(n_clusters=self.k, random_state=0).fit(pixels)
        centers = np.uint8(kmeans.cluster_centers_)
        labels = kmeans.labels_
        counts = np.bincount(labels)
        dominant = centers[np.argmax(counts)]
        return dominant, centers, counts
    def simulate_image(self, width=100, height=100):
        image = np.zeros((height, width, 3), dtype=np.uint8)
        for i in range(height):
            for j in range(width):
                image[i, j] = [random.randint(0, 255) for _ in range(3)]
        return image

def main():
    cs = CloudStorageModule()
    ac = AccessControlModule()
    aa = AudioAnalysisModule()
    vqc = VideoQualityController()
    cr = ColorRecognitionModule()
    cs_files = cs.simulate_large_upload(10)
    ac_results = ac.simulate_access_control()
    audio_peaks = aa.analyze_audio(2)
    video_qualities = vqc.simulate_network_conditions(15)
    image = cr.simulate_image()
    dominant_color, centers, counts = cr.extract_dominant_colors(image)
    print("Cloud Files:", cs_files)
    print("Access Control Results:", ac_results)
    print("Audio Peaks Frequencies:", audio_peaks[0])
    print("Video Qualities:", video_qualities)
    print("Dominant Color:", dominant_color)
    print("KMeans Centers:", centers)
    print("Cluster Counts:", counts)

if __name__ == "__main__":
    main()
```