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 студентами під час тестування для контролю рівня знань”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Поданьов А.О.
«____» _____ 2025 р.

Керівник проекту
доктор технічних наук, професор
_____ Коваленко О.В.
«____» _____ 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

Поданьову Артуру Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи відеоспостереження за студентами під час тестування для контролю рівня знань*

2. Керівник роботи *Коваленко Олександр Володимирович, докт. техн. наук, професор*
(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 студентами під час тестування для контролю рівня знань*

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
_____ Коваленко О.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Поданьов А.О.
(прізвище та ініціали)

АНОТАЦІЯ

Поданьов А.О. Програмне забезпечення системи відеоспостереження за студентами під час тестування для контролю рівня знань. 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 студентами під час тестування для контролю рівня знань.

Метою розробки є програмне забезпечення системи відеоспостереження за студентами під час тестування для контролю рівня знань.

Результат роботи – програмна реалізація системи відеоспостереження за студентами під час тестування для контролю рівня знань.

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, тестування для контролю рівня знань

ABSTRACT

Podanov A.O. Software for a video surveillance system for students during testing to control the level of knowledge. 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, which is intended for a video surveillance system for students during testing to control the level of knowledge.

The purpose of the development is software for a video surveillance system for students during testing to control the level of knowledge.

The result of the work is a software implementation of a video surveillance system for students during testing to control the level of knowledge.

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 PCs with Windows 10/11.

The program is developed in the Python environment.

Keywords: computer engineering, video surveillance, testing for knowledge control

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ВКС	–	відеоконференцзв’язок
ПЗ	–	програмне забезпечення
CMP	–	Codian Management Platform
DNS	–	Domain Name System
GMS	–	Global Management System
GRE	–	Generic Routing Encapsulation
ISDN	–	Integrated Services Digital Network
MCU	–	Multipoint Control Unit
MPLS	–	Multiprotocol Label Switching
MXM	–	Media eXchange Manager
SSL	–	Secure Socket Layer
TLS	–	Transport Layer Security
TMS	–	Tandberg Management Suite
VPN	–	Virtual Private Network
USENET	–	User Network

ВСТУП

Актуальність теми. У останній час, у зв'язку з переходом на нові технології, у закладах вищої освіти (ЗВО) запроваджуються новітні технології навчання. До цих технологій можливо віднести різні методи дистанційного контролю якості знань, у тому числі й під час тестування, за допомогою різного плану систем відеоспостереження.

Багато навчальних закладів звертаються до передових технологічних рішень, таких як системи відеоспостереження, щоб підвищити безпеку кампусу.

При цьому існує політика університетської відеобезпеки, яка забезпечує безпеку студентів і майна ЗВО.

Давайте обговоримо важливість таких політик, їх ключові компоненти та найкращі практики для забезпечення законного та етичного використання технологій стеження в освітніх установах.

Політика університетського відеоспостереження розроблена для забезпечення законного та професійного використання камер відеоспостереження відповідно до університетських стандартів безпеки. Він відображає, як і коли уповноважена особа може контролювати та контролювати системи відеоспостереження.

Політика повинна включати розумні процедури збереження даних, пов'язаних із будь-яким інцидентом, у тому числі порушенням університетських стандартів безпеки. Загалом політика університетського відеоспостереження повинна охоплювати питання конфіденційності, справедливого розкриття інформації та належного використання систем спостереження.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи відеоспостереження за студентами під час тестування для контролю рівня знань.

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

– Огляд існуючих систем відеоспостереження за студентами під час тестування для контролю рівня знань.

– Дослідження системи відеоспостереження за студентами під час тестування для контролю рівня знань.

– Програмна реалізація системи відеоспостереження за студентами під час тестування для контролю рівня знань.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі відеоспостереження за студентами під час тестування для контролю рівня знань.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи відеоспостереження за студентами під час тестування для контролю рівня знань, 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

У кожній області має бути встановлена політика університетських камер відеоспостереження, щоб забезпечити ефективне використання камер відеоспостереження на кампусі. У цій політиці має бути чітко зазначено, що відзнятий матеріал може бути представлений співробітникам правоохоронних органів для цілей розслідування.

Ефективна політика університетських камер відеоспостереження має гарантувати конфіденційність студентів та іншого університетського персоналу та відповідати загальним стандартам безпеки ЗВО.

Усі камери відеоспостереження, на які поширюється ця політика, будуть перераховані в базі даних, яку ведуть органи вищого рівня. Будь-які існуючі камери безпеки або системи спостереження повинні бути приведені у відповідність до цієї політики. Щоб контролювати несанкціоноване використання камер безпеки, їх потрібно розмістити в безпечному місці та налаштувати для запобігання несанкціонованому доступу, дублюванню або модифікації. Але чи незаконно мати камери в університетських класах?

Щодо політики щодо університетських камер відеоспостереження, інформування студентів, вчителів і батьків про процедури та погодження в рамках цього положення є обов’язковим. Має бути зрозуміло, що відео- та аудіо, записані університетськими камерами відеоспостереження, можуть використовуватися як доказ для допомоги у розслідуванні.

Студенти, адміністратори кампусу та персонал можуть бути сповіщені через видимі вивіски про те, що "будівлі та території університетського округу обладнано електронним спостереженням для безпеки студентів, персоналу та відвідувачів кампусу. Ваші дії можуть бути записані та збережені".

1.2 Область застосування

Областю застосування системи є заклади вищої освіти. Що повинна охоплювати політика університетського спостереження?

Вирішіть питання конфіденційності

Політика університетського відеоспостереження не повинна дозволяти спостереження за приватними зонами або зонами, де є розумне очікування приватності. Ці зони можуть включати туалети, роздягальні або роздягальні.

Нікому не повинно бути дозволено записувати конфіденційну інформацію без згоди сторін. Відповідні процедури університету серйозно розглядатимуть будь-які порушення цієї політики.

Авторизоване використання відеокамер спостереження

Контрольований та обмежений доступ до камер спостереження має бути головним пріоритетом університетської політики відеоспостереження. Тільки уповноважений персонал, визначений школою, повинен мати доступ до камер відеоспостереження та записів.

Виходячи зі стандартів безпеки ЗВО, захищений паролем доступ повинен бути обмежений окремим персоналом ЗВО або менеджерами з безпеки. Політика має зберігати конфіденційність записаного матеріалу від широкого загалу та поважати конфіденційність окремих осіб.

Уникайте непотрібних вторгнень

Згідно з правилами університетського відеоспостереження, щоб уникнути непотрібних вторгнень, необхідно використовувати охорону та камери спостереження. Не повинно бути втручання в академічну свободу студентів, приватне життя чи свободу слова. Це зрештою допоможе у професійному та етичному використанні камер спостереження.

Відповідне та заборонене використання

Політика університетського відеоспостереження повинна забороняти неавторизованим особам або групам доступу до камер безпеки та записів. Дані,

що зберігаються в камерах, не повинні бути доступні або розголошені нікому, крім випадків, визначених політикою.

Лише уповноваженим особам має бути дозволено використовувати камери спостереження для спостереження за людьми в кампусі та навколо нього.

Зберігання записаного матеріалу

Відповідно, зберігання та оприлюднення записаного матеріалу є дуже важливим. Політика може допомогти розробити процедури збереження записаного матеріалу в безпеці та надійності відповідно до університетських стандартів безпеки.

Оприлюднення записаного матеріалу

Правила безпеки дозволяють оприлюднювати інформацію лише особам, визнаним політикою ЗВО. Важливо зауважити, що оприлюднення інформації не повинно порушувати конфіденційність особи. Слід оприлюднити лише необхідні матеріали, зберігаючи конфіденційність.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи відеоспостереження за студентами під час тестування для контролю рівня знань, 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

ЗВО служать меті, що виходить за межі їх основної функції як місця для навчання, як спеціалізовані установи, покликані формувати мислення та характери наступного покоління. Сучасні ЗВО також забезпечують безпечний притулок для студентів завдяки технологічним досягненням надійного відеоспостереження.

Відеоспостереження в ЗВО стало невід'ємною частиною протоколів безпеки на тлі зростаючої стурбованості безпекою в суспільстві. Камери відеоспостереження, які також називаються університетськими камерами безпеки, пропонують комплексне та динамічне рішення, яке забезпечує дотримання правил, стежить за рухами та визначає потенційні загрози безпеці мешканців.

Впровадження відеоспостереження в ЗВО передбачає ретельний, систематичний нагляд за приміщеннями, перетворюючи навчальні заклади на безпечне середовище навчання. Керівники закладів все частіше встановлюють ці пристрої безпеки в державних, приватних і чартерних ЗВО. Ці пристрої мають вбудовані функції, які відлякують такі загрози, як несанкціоновані проникнення та інші ймовірні злочинні дії. Камери відеоспостереження також гарантують, що студенти дотримуються інституційних інструкцій і дисципліни.

Спостереження в навчальних закладах стало свідком значного технологічного прогресу, який узгоджується із загальною революцією безпеки та спостереження в інших галузях. Хоча традиційні системи відеоспостереження колись відігравали вирішальну роль у забезпеченні безпеки в ЗВО, технологічний

прогрес проклав шлях до розумніших систем відеоспостереження з підвищенням ефективності, надійності та продуктивності завдяки операціям у реальному часі.

Система камер штучного інтелекту надає вдосконалені рішення для відеоспостереження з високотехнологічними рішеннями для підвищення безпеки об'єктів і підвищення ефективності роботи в ЗВО.

Гнучкість системи камер зі штучним інтелектом дозволяє ЗВО легко інтегрувати свою систему камер із такими важливими інтеграціями, як контроль доступу до дверей, датчики навколишнього середовища, виявлення сигарет тощо. Ці інтелектуальні інтеграції підсилюють стратегії безпеки з мінімальною суєтою та адаптацією.

Сучасне відеоспостереження в ЗВО, інтегроване штучним інтелектом, є величезним кроком у безпеці ЗВО. Системні технологічні камери безпеки для ЗВО підвищують заходи безпеки та сприяють новітній ефективності та узгодженості, революціонізуючи стандарти безпеки в сучасних навчальних закладах.

Впровадження технології безпеки в ЗВО

ЗВО значно вдосконалили свої стратегії безпеки завдяки зростаючій кількості повідомлень про інциденти в ЗВО, які включають насильницькі та ненасильницькі інциденти, залякування, куріння, вживання алкоголю тощо.

Таким чином, університетські спеціалісти з ресурсів, ІТ-команди та адміністрація ЗВО стали свідками підвищення попиту на більш просунуті рішення безпеки кампусу для оптимізації безпеки студентів, викладачів і персоналу.

Щоб забезпечити безпеку студентів, навчальні заклади звертаються до впровадження рішень на основі штучного інтелекту, щоб отримати кращу видимість у кампусі, а також знімати тягар з внутрішніх команд. Завдяки системі камер штучного інтелекту ЗВО можуть:

– Швидко налаштуйте свою нову систему, використовуючи майже будь-яку IP-камеру, яку вони бажають.

– Об’єднайте всі камери на одній сучасній, зручній панелі керування.

– Використовуйте штучний інтелект для швидшого пошуку відеозаписів і вирішення інцидентів.

– Інтегруйте свою систему камери з найважливішими інструментами, такими як контроль доступу або датчики вейпінгу.

Правильна система камер зі штучним інтелектом для ЗВО надає ці можливості завдяки інтелектуальному відеореєстратору (IVR), який працює з найновішими графічними процесорами NVIDIA як мережевий пристрій, який підтримує надійне локальне зберігання, доповнене перевагами хмари. Цей тип системи відеоспостереження також має хмарну інформаційну панель, яка забезпечує видимість зареєстрованих ЗВО і камер через єдину платформу.

Інтелектуальні камери відеоспостереження забезпечують глибоку видимість у університетських приміщеннях, підвищуючи безпеку різними способами. Ці вдосконалені установки запобігають незаконній діяльності на території ЗВО, щоб допомогти персоналу служби безпеки швидко реагувати на потенційні небезпеки та записувати важливі та оперативні докази для складання карт і розслідування інцидентів.

Зважений підхід до безпеки ЗВО

Незважаючи на їх різноманітні переваги, університетські камери спостереження викликають негативні відгуки. Дехто припускає, що постійне спостереження може порушити навчальне середовище, створюючи у студентів і вчителів відчуття постійного контролю. Таким чином, ці пристрої, можливо, можуть поставити під загрозу конфіденційність відвідувачів ЗВО.

Незважаючи на ці побоювання, ЗВО повинні зберігати відповідальність за оптимізацію безпеки своїх студентів і персоналу за допомогою найкращих практик безпеки в ЗВО. Для керівників закладів залишається життєво важливим впровадження найкращих практик безпеки в ЗВО, наприклад розгортання вдосконаленого відеоспостереження.

Ці пристрої мають забезпечувати надійне покриття відеоспостереження,

дотримуючись індивідуальних інструкцій щодо конфіденційності. Збалансований підхід до запровадження систем відеоспостереження в ЗВО міг би включати ретельну орієнтацію та курси підвищення кваліфікації щодо протоколів безпеки, прозорість розташування камер, а також відкрите обговорення проблем конфіденційності та рішень.

Spot AI

Обмеження розміщення камер спостереження в місцях з інтенсивним рухом людей, таких як коридори, кафетерії чи доріжки, може зняти проблеми з конфіденційністю. Spot AI стоїть в авангарді дебатів про безпеку, створюючи складні системи відеоспостереження для підвищення безпеки кампусу та заходів безпеки студентів, одночасно вирішуючи останні проблеми конфіденційності.

Використовуючи таку передову технологію безпеки в ЗВО, викладачі можуть посилити свої заходи безпеки та підтримувати безпечніше середовище, яке сприяє найкращим результатам навчання для кожного учня.

Вибираємо найкращу систему відеоспостереження для ЗВО

Поява систем відеоспостереження для ЗВО стала невід'ємною частиною задоволення зростаючих вимог до безпеки університетських будівель. Відеоспостереження є важливим інструментом, який забезпечує безпеку студентів, викладачів і персоналу, надаючи повну інформацію про інциденти, що відбуваються всередині та навколо університетських приміщень.

Удосконалені системи університетських камер пропонують безліч високопродуктивних функцій, які забезпечують стандартні університетські стратегії безпеки ефективністю сучасних технологій. Наприклад, ці системи відеоспостереження можуть містити інтелектуальну відеоаналітику.

Кампусна відеоаналітика представляє оцінку локальних сценаріїв у реальному часі за допомогою уточнених, детальних і вдосконалених даних спостереження. Платформа Spot AI знаходиться в авангарді цих інновацій у системі спостереження з системою, яка постійно покращує ефективність моніторингу безпеки, аналізуючи шаблони та незвичайні дії в різних контекстах.

Інтелектуальна платформа безперебійно працює з будь-якою конструкцією IP-камери, забезпечуючи безпроблемну інтеграцію для ЗВО, які вважають за краще зберегти наявне обладнання для відеоспостереження. Керівники закладів також можуть вибрати безкоштовне оновлення IP-камери Spot AI. Зручний підхід Spot AI до технології спостереження оптимізує застарілу інфраструктуру, водночас забезпечуючи ефективність безпеки в майбутньому.

Інтелектуальний відеореєстратор (IVR) від Spot AI є однією з важливих функцій передової системи відеоспостереження. IVR на основі найновіших графічних процесорів NVIDIA представляє оптимізовану конструкцію для зручного використання пропускної здатності та цілодобового локального зберігання даних поряд із перевагами хмари. Гібридна конфігурація забезпечує постійне безперебійне спостереження та доступ до даних критично важливого матеріалу для легкого перегляду, аудиту та розслідування безпеки.

Spot AI також надає хмарну інформаційну панель – централізовану точку дотику, яка відображає канали з кожної встановленої камери, розташованої на території ЗВО. Користувачі можуть зручно отримувати доступ до інформаційної панелі з комп’ютерів, мобільних телефонів або програм.

Універсальна інформаційна панель покращує досвід відеоспостереження, пропонуючи універсальне рішення, яке оптимізує відео спостереження з багатьох місць за допомогою єдиної платформи. Окрім загальних рішень для відеоспостереження, ЗВО також дедалі більше визнають важливість детекторів для сигарет. Вейп-детектори для ЗВО є необхідним доповненням для підтримки безпеки та здоров’я студентів у відповідь на зростання вейпінгу серед підлітків.

Ці високочутливі університетські детектори вейпа можуть попередити владу про ознаки вейпінгу, щоб допомогти стримати поширення нездорових звичок у ЗВО. Зокрема, детектори вловлюють мікрочастинки (на основі хімічного розпаду) з навколишнього середовища, швидко й точно повідомляючи адміністраторів про випадки використання сигарет.

Удосконалена система відеоспостереження Spot AI також пропонує

контроль доступу до ЗВО, що особливо ефективно для обмеження доступу до приміщень із високим рівнем безпеки для авторизованого персоналу. Критична функція спостереження покращує систему безпеки ЗВО, запобігаючи порушенням, які ставлять під загрозу безпеку мешканців.

Зрештою, передові технології університетської безпеки та спостереження пропонують комплексні рішення для різноманітних проблем, з якими стикається адміністрація ЗВО. Камери спостереження в ЗВО забезпечують постійний моніторинг, який інформує адміністраторів про активні кроки, необхідні для підтримки оптимальної безпеки будівлі.

Стратегічне поєднання відеоспостереження, відеоаналітики університетського містечка, детекторів електронних сигарет і контролю доступу підвищує безпеку в університетських приміщеннях, одночасно сприяючи ефективності роботи. Комплексна система відеоспостереження Spot AI для ЗВО може надати навчальним закладам надійний і сучасний підхід до безперервного підвищення безпеки, який забезпечить спокій кожному учню, викладачу та відвідувачу.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – динамічна інтерпретована об’єктно-орієнтована скриптова мова програмування із строгою динамічною типізацією. Офіційний сайт мови програмування Python https://www.python.org/. Python – багатоцільова мова програмування, яка дозволяє писати код, що добре читається. Відносний лаконізм мови Python дозволяє створити програму, яка буде набагато коротше свого аналога, написаного на іншій мові. Python – багатоплатформова мова програмування. Це означає, що програми на Python можна запускати в різних операційних системах без будь-яких змін.

Ще однією перевагою Python є його стандартна бібліотека, яка встановлюється разом з Python і містить готові інструменти для роботи з операційною системою, веб-сторінками, базами даних, різними форматами даних, для побудови графічного інтерфейсу програм тощо. Програми, написані на мові програмування Python, можуть бути як невеликими скриптами, так і складними системами. Python абсолютно безкоштовний.

Швидкість виконання коду Python

Один з можливих недоліків Python – швидкість виконання коду. Python не є компільованою мовою. Код на Python спочатку компілюється у внутрішній байт-код, який потім виконується інтерпретатором Python. У більшості випадків при використанні Python виходять програми повільніші в порівнянні з такими мовами, як C.

Втім, сучасні комп'ютери мають таку обчислювальну потужність, що для більшості застосунків швидкість розробки важливіша швидкості виконання, а програми на Python зазвичай пишуться набагато швидше.

Окрім того, Python легко розширюється модулями, написаними на C або C++. Такі модулі можуть використовуватися для виконання частин програми, що створюють інтенсивне навантаження на процесор.

Використання Python

Python використовується для різних цілей: для створення ігор і веб-застосунків, розробки внутрішніх інструментів для різноманітніх проектів. Мова також широко застосовується в науковій області для досліджень і розв'язування прикладних завдань.

Застосування мови програмування Python:

1. BitTorrent – протокол для обміну даними.

2. Ubuntu Software Center – вільне програмне забезпечення для пошуку, установки і видалення пакунків в системі Ubuntu Linux.

3. Blender – програма для створення тривимірної комп'ютерної графіки, що включає засоби моделювання, анімації, вимальовування, пост-обробки відео, а також створення відеоігор.

4. GIMP – растровий графічний редактор, із підтримкою векторної графіки.

5. World of Tanks.

6. Вільна енциклопедія Вікіпедія.

7. Пошукова система Google.

8. DropBox – файловий хостинг, що включає персональне хмарне сховище, синхронізацію файлів і програму-клієнт.

9. YouTube – популярне відеосховище.

Версії Python

Мови програмування з часом змінюються – розробники додають в них нові можливості, а також виправляють помилки. Так з'являються різні версії мови. Наприклад, код написаний на Python 2 у більшості випадків не буде працювати у версії Python 3 без внесення додаткових змін.

Процесор є найважливішим компонентом в комп'ютері. Одна з основних функцій процесора – це обробка даних згідно комп'ютерної програми, яка є списком інструкцій, шляхом виконання арифметичних і логічних операцій над фрагментами даних.

Кожна інструкція в програмі – це команда, яка «повідомляє» процесору, яку операцію він повинен виконати. Процесор комп'ютера може розуміти лише ті інструкції, які написані на машинній мові. Машинна мова – це штучна мова, створена для передачі команд комп'ютеру. За допомогою машинної мови створюються ефективні програми, оскільки розробник отримує доступ до всіх можливостей процесора. Машинна мова – мова низького рівня.

Інструкція машинної мови існує для кожної операції, яку процесор здатний виконати – є інструкція для додавання чисел, є інструкція для віднімання

чисел і т.д. Увесь набір інструкцій, який центральний процесор може виконати, відомий як набір інструкцій процесора.

Наприклад, у вас є певна програма, яка зберігається на диску вашого комп'ютера. Для виконання програми, ви здійснюєте подвійний клік на значку програми. Це змушує програму копіюватися з диска в оперативну пам'ять, після чого процесор комп'ютера виконує копію програми, яка знаходиться в оперативній пам'яті.

Коли процесор виконує інструкції програми, він бере участь у процесі, який є відомим як цикл fetch – decode – execute (отримати – декодувати – виконати). Цей цикл виконується для кожної інструкції у програмі і складається з трьох кроків:

Отримати

Програма – це послідовність інструкцій на машинній мові. Першим кроком циклу є завантаження (отримання) наступної інструкції з пам'яті в процесор.

Декодувати

Інструкція машинної мови – це двійкове число, яке представляє команду, що повідомляє процесору виконати певну операцію. На цьому кроці процесор декодує інструкцію, яку було «витягнуто» з пам'яті, для визначення того, яка операція повинна виконуватись.

Виконати

Останній крок циклу – виконати операцію.

Хоча процесор комп'ютера розуміє тільки машинну мову, людині непрактично писати програми на машинній мові. Така програма може мати тисячі або навіть мільйони бінарних інструкцій, і написання такої програми буде дуже обтяжливим процесом.

З цієї причини була створена мова асемблера як альтернатива машинній мові. Замість використання двійкових чисел для написання інструкцій, мова асемблера використовує короткі слова, відомі як мнемокоди.

Незважаючи на те, що мова асемблера не вимагає двійкових інструкцій, як у випадку машинної мови, проте вона вимагає високих знань про процесор. Використовуючи мову асемблера, навіть для найпростішої програми, необхідно написати велику кількість інструкцій.

Мова програмування високого рівня дозволяє створювати складні програми, не знаючи, як працює процесор, і не записуючи великої кількості інструкцій низького рівня. Крім того, більшість мов програмування високого рівня використовують слова, які легко зрозуміти.

Python – одна із популярних сучасних мов програмування високого рівня. Python – інтерпретована мова програмування. Python – це високорівнева інтерпретована мова програмування, на відміну від C++, яка є прикладом компільованої мови програмування. Назва Python відноситься як до мови програмування, так і до інтерпретатора – комп'ютерної програми, яка зчитує початковий код (написаний на Python) і виконує інструкції (команди).

Для перекладу мови високого рівня на машинну мову доступні два типи програм:

1. Компілятор.

2. Інтерпретатор.

Завантаження Python

Версії інтерпретатора Python для різних операційних систем доступні для безкоштовного завантаження за адресою https://www.python.org/downloads.

Середовище програмування для Python

Для написання програм використовують текстові редактори або інтегровані середовища розробки, які включають в себе різні інструменти для роботи з кодом: засіб для написання коду (текстовий редактор), інтерактивний інтерпретатор, відлагоджувач тощо.

Текстові редактори та інтегровані середовища програмування для Python:

– IDLE – стандартний редактор Python. Встановлюється разом з Python для користувачів Windows, окремим пакунком для користувачів Linux.

– Notepad++ – безкоштовний текстовий редактор початкового коду, який підтримує велику кількість мов, в тому числі і Python. Лише для користувачів Windows.

– Visual Studio Code – це легкий, але потужний редактор початкового коду, який розповсюджується безкоштовно і доступний у версіях для платформ Linux, Windows і macOS.

– PyScripter – інтегроване середовище розробки для мови програмування Python. Для користувачів Windows. Поширюється безкоштовно.

– Wing IDE 101 – вільне інтегроване середовище для Python, розроблене для навчання програмістів-початківців. Для користувачів Linux, Windows і macOS. Поширюється безкоштовно.

– Geany – вільний текстовий редактор з базовими елементами інтегрованого середовища розробки, доступний для операційних систем Linux, Windows і macOS.

– PyCharm – інтегроване середовище розробки для мови програмування Python. PyCharm є власницьким програмним забезпеченням. Наявна безкоштовна версія Community з усіченим набором можливостей. Для користувачів Linux, Windows і macOS.

– Thonny – IDE для вивчення програмування мовою Python. Для користувачів Linux, Windows і macOS.

– Mu – редактор коду Python для програмістів-початківців. Для користувачів Linux, Windows і macOS.

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 студентами під час тестування для контролю рівня знань.

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

Стандартні завдання, які ставляться перед системою відеоспостереження

Стандартні завдання, що коштують перед відеоспостереженням на будь-якому об'єкті, будь те великий, малий бізнес або приватний будинок, схожі. Виділимо основні:

1. Поточне спостереження.

2. Робота з архівом відеозаписів.

3. Дистанційний перегляд поточне зображення й архіву.

4. Запис відеозображення по детекторі руху, а також при спрацьовуванні охоронних датчиків або втраті сигналу.

На великому об'єкті до стандартного додаються наступні завдання:

1. Інтеграція із системою охоронної й пожежної сигналізації.

2. Інтеграція з апаратно-програмним комплексом системи контролю й управління доступом (СКУД).

3. Масштабованість і модернізація системи відеоспостереження при необхідності.

4. Поточне спостереження й управління всією системою з однієї точки, у тому числі організація відеоспостереження через Інтернет.

Для роботи із системами відеоспостереження необхідно програмне забезпечення (ПЗ), що дозволяє найбільше комфортно й доступно використовувати всього функціонала відеокамер для користувача. На жаль, стандартний софт, яким оснащені пристрої й компоненти відеосистеми, не завжди відповідає даним вимогам.

До основних систем керування системи відеоспостереження за студентами під час тестування для контролю рівня знань відносять:

– Tandberg Management Suite – TMS.

– Global Management System – GMS; PathNavigator.

– Media eXchange Manager – MXM.

– Codian Management Platform – CMP; Codian Director; Codian Scheduller.

Оскільки ЗВО використовують камери відеоспостереження в різних місцях на кампусі та навколо нього з метою безпеки, їм також потрібна політика відеоспостереження. Це необхідно для забезпечення професійного використання камер відеоспостереження. Якщо ЗВО використовують камери відеоспостереження, вони повинні дотримуватися кодексів поведінки, проілюстрованих у політиці.

Крім того, у політиці відеоспостереження має бути чітко визначено, чи записуються особи чи ні, і де вони записуються. Політику відеоспостереження можна використовувати для охорони та захисту кампусу та осіб у приміщеннях. Крім того, його можна використовувати для збору будь-яких істотних доказів, необхідних для підтримки розслідувань.

У ЗВО є законним спостерігати за студентами на камерах, якщо вони не порушують їхнє приватне життя. Камери безпеки та системи спостереження не слід встановлювати там, де загальноприйняті соціальні норми потребують конфіденційності.

Доступ до записів та інформації з камер спостереження суворо обмежений уповноваженим персоналом. Це можуть бути співробітники, треті особи, які подали запит на вхід, батьки та повістки.

Наскільки важливим є доступ і зберігання камер відеоспостереження в ЗВО , їхнє розміщення також є дуже важливим.

Згідно з правилами університетських камер відеоспостереження, спостереження може здійснюватися біля головних входів, виходів із кампусу,

складських приміщень, камер схову, автостоянок, конференц-залів, ігрових майданчиків, класів, коридорів, місць відпочинку та адміністративних кафетеріїв.

Проте камери відеоспостереження повинні встановлюватися та використовуватися суворо для забезпечення безпеки людей і адміністрації.

Є місця, де використання камер відеоспостереження вкрай неприпустимо. До них відносяться туалети, приватні спальні, камери схову, роздягальні та кабінети персоналу.

Камери відеоспостереження використовуються виключно для забезпечення безпеки та моніторингу ЗВО. Розташування камер має внести адміністратор. Будь-яке розміщення, яке порушує конфіденційність особи, є неприйнятним і може бути підставою для порушення.

Операторів університетського відеоспостереження навчать використовувати камери законно та етично. Вони будуть проінформовані про технічне використання камер спостереження. Вони визнають, що розуміють політику університетського відеоспостереження.

Оператори будуть виконувати відповідальність і політику. Крім того, оператори відеоспостереження матимуть доступ до записів, якщо це буде дозволено політикою.

Хоча політика університетського відеоспостереження дозволяє операторам переглядати відзнятий матеріал для покращення заходів безпеки, деякі заборони залишаються. Оператори відеоспостереження не можуть стежити за студентами на основі особистих характеристик, таких як раса, стать, етнічна приналежність чи інвалідність.

Їм також заборонено стежити за будь-якою інтимною поведінкою або використовувати будь-які інші системи відеоспостереження чи зображення спостереження, які не відповідають цій політиці.

Області повинні забезпечити добре сформовану політику університетського відеоспостереження відповідно до університетських стандартів

безпеки. Політика розробляється з урахуванням обмежень конфіденційності окремих осіб.

Настійно рекомендується розміщувати вивіски біля входу та виходу з кампусу. Студенти також можуть отримувати інформацію через інформаційні бюлетені та довідники для співробітників.

Якщо для підтвердження запитується запис, його слід зберігати в безпеці, доки прийнятні стандарти не розглянуть запит. Рекомендуємо зберегти копію розкриття інформації для майбутніх застережень.

Комплексна політика університетського відеоспостереження має вирішальне значення для підтримки безпечного та надійного навчального середовища.

Збалансовуючи потребу в безпеці з турботою про конфіденційність, ЗВО можуть ефективно запобігати неправомірній поведінці, швидко реагувати на інциденти та забезпечувати благополуччя студентів, викладачів, викладачів і відвідувачів кампусу.

Завдяки чітким інструкціям щодо розміщення камер, зберігання даних, контролю доступу та дотримання правил, чітко визначена політика спостереження сприяє прозорості, підзвітності та довірі в університетській спільноті.

3.2 Розробка структурної схеми

Структурна схема розробленого, у ході виконання дипломного проектування, програмного забезпечення зображена на рисунку 3.1.

Програмне забезпечення обробки відеоінформації з Web-камери по мережі Інтернет складається з наступних структурних блоків:

– Аналогові камери.

– Web-сервери (V1netServer510).

– IP-камери встановлені усередині приміщення (V1netIP220; V1netIP230;

V1netIP250).

– Мережний комутатор (TCP/IP).

– Точки доступу Wi-Fi IEEE 802.11b/g (10 Мбіт/с).

– Маршрутизатор або модем для зв'язку з віддаленими робітниками місцями через Інтернет.

– Розроблене, у ході виконання дипломного проектування, програмне забезпечення на мережних робочих місцях.

– Розроблене, у ході виконання дипломного проектування, програмне забезпечення на віддалених робочих місцях (доступ через Інтернет).

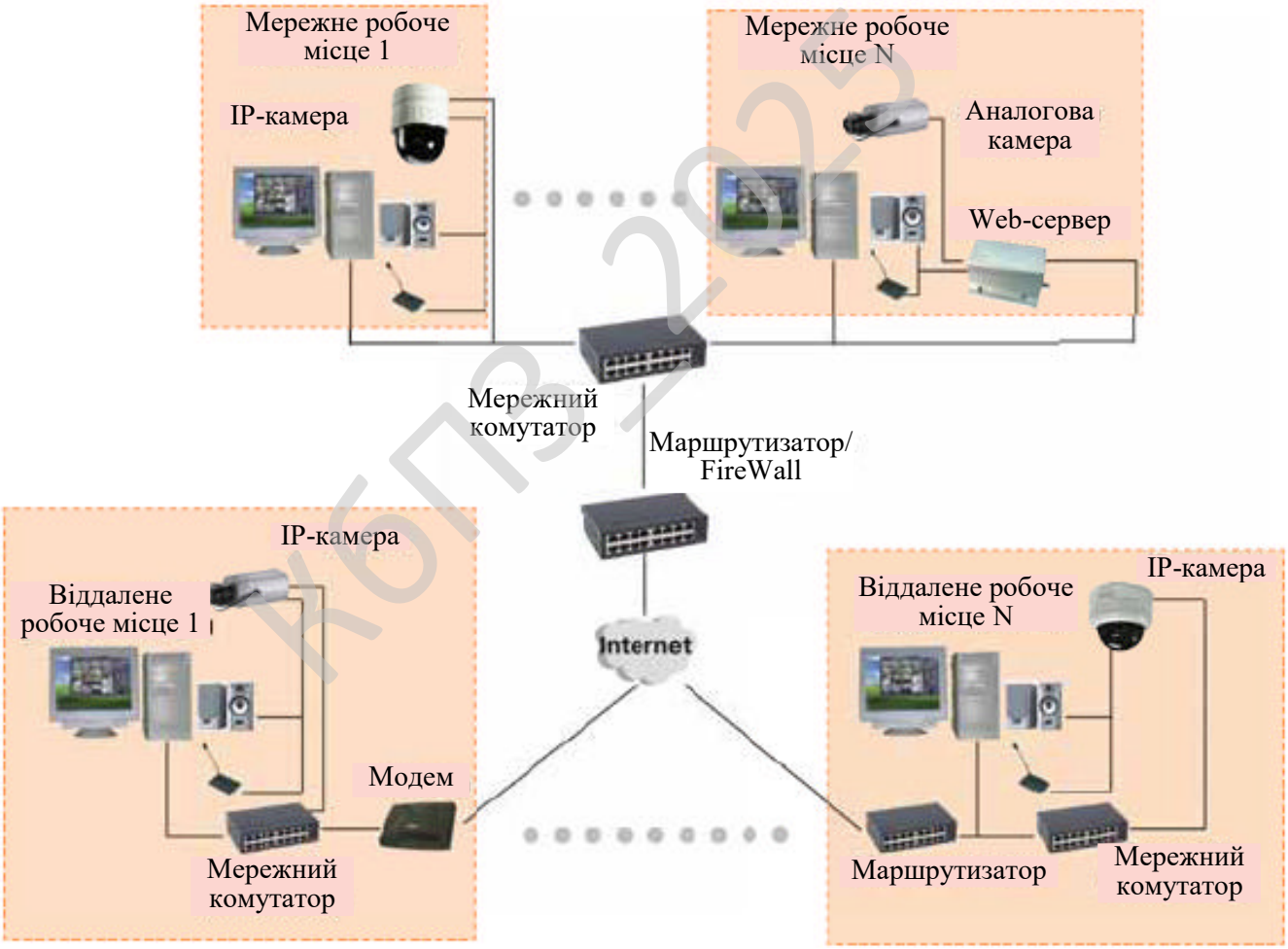


Рисунок 3.1 – Структурна схема системи

Для розрахунку параметрів обробки відеоінформації з Web-камери по мережі Інтернет, можна використовувати дані наведені в таблицях 3.1-3.3.

Таблиця 3.1 – Розрахунок пропускної здатності ЛОМ

Параметри налаштування			Кількість IP-камер / Пропускна здатність ЛОМ					
Розрішення	кадр/с	кбіт/с	1	4	9	16	32	64
D1 720x576	25	1500	1,5M	6,0M	13,5M	24,0M	48,0M	96,0M
	12	750	750K	3,0M	6,8M	12,0M	24,0M	48,5M
	8	512	512K	2,0M	4,6M	8,2M	16,4M	32,8M
	5	256	256K	1,0M	2,3M	4,1M	8,2M	16,4M
	1	56	56K	0,2M	0,5M	0,9M	1,8M	3,6M
CIF 352x288	25	750	750K	3,0M	6,8M	12,0M	24,0M	48,0M
	12	384	384K	1,5M	3,5M	6,1M	12,3M	24,6M
	8	256	256K	1,0M	2,3M	4,1M	8,2M	16,4M
	5	128	128K	0,5M	1,2M	2,0M	4,1M	8,2M
	1	28	28K	0,1M	0,3M	0,4M	0,9M	1,8M

У стовпці «Параметри налаштування», зазначена рекомендована швидкість передачі в кбіт/с.

При збільшенні швидкості передачі, якість зображення збільшується, при зменшенні – якість зображення знижується.

Максимальна кількість одночасних клієнтських підключень розраховує калькулятор у меню налаштувань IP-камер (серверів).

Дані, які представлені у таблиці 3.2 допомагають здійснити розрахунок обсягу жорсткого диска для відеоархіву (на один канал).

У таблиці 3.2 зазначені середні значення, які залежать від інтенсивності руху в кадрі (алгоритм компресії MPEG-4).

Таблиця 3.2 – Розрахунок обсягу жорсткого диска для відеоархіву (на один канал)

Параметри налаштування			Тривалість запису / Обсяг жорсткого диска					
Розрішення	кадр/с	кбіт/с	1 сек.	1 година	1 день	14 дн.	30 дн.	60 дн.
D1 720x576	25	1500	185,5 K	667,8 M	16,2 G	226,8 G	486,0 G	972,0 G
	12	750	93,8 K	337,7 M	8,1 G	113,5 G	243,1 G	486,0 G
	8	512	64,0 K	230,4 M	5,5 G	77,4 G	165,9 G	331,8 G
	5	256	32,0 K	115,2 M	2,8 G	38,7 G	82,9 G	165,9 G
	1	56	7,0 K	25,2 M	0,6 G	8,5 G	18,1 G	36,3 G
CIF 352x288	25	750	93,8 K	337,7 M	8,1 G	113,5 G	243,1 G	486,3 G
	12	384	48,0 K	172,8 M	4,1 G	58,1 G	124,4 G	248,8 G
	8	256	32,0 K	115,2 M	2,8 G	38,7 G	82,9 G	165,9 G
	5	128	16,0 K	57,6 M	1,4 G	19,4 G	41,3 G	82,9 G
	1	28	3,5 K	12,6 M	0,3 G	4,2 G	9,1 G	18,1 G

Вхідні дані:

– Число кадрів у день: 16000.

– Середня довжина кадру: 1,275.

– Тривалість дня в годинах: 8.

3.3 Розробка функціональної схеми

Багатоточечний сервер системи відеоспостереження за студентами під час тестування для контролю рівня знань, разом із системою керування є гнучким модульним багатофункціональним засобом для підтримки багатоточечних систем відеоспостереження за студентами під час тестування для контролю рівня знань. Функціональна схема сервера системи відеоспостереження за студентами під час тестування для контролю рівня знань наведена на рисунку 3.2.

Крім модулів каналів зв'язку, сервер системи відеоспостереження за студентами під час тестування для контролю рівня знань, містить модулі звукового процесора й відеопроцесора, синхронізатора зображення й звуку, транскодера H.261/H.263, а також набір інтерфейсних модулів, зв'язаних системною шиною. Керуючі модулі керують роботою всієї системи. На «транковому» рівні сервер системи відеоспостереження за студентами під час тестування для контролю рівня знань, звичайно підтримує інтерфейси IDNX-PRI (Integrated Services Digital Network Primary Rate Interface) і T1 (E1) BBS (Robbed-Bit Signaling), на лінійному рівні – інтерфейс ISDN-BRI (Integrated Services Digital Network Basic Rate Interface), протокол цифрових з'єднань DCP (Digital Communications Protocol) і аналогові лінійні канали для з'єднання через модеми. Звичайно сервер системи відеоспостереження за студентами під час тестування для контролю рівня знань, заснований на стандартах H.320 і T.120 MCE-T (Сектора T Міжнародного союзу електрозв'язку) і має широкі можливості транскодування, що забезпечує сумісність його з найрізноманітнішими засобами системи відеоспостереження за студентами під час тестування для контролю рівня знань. В таблиці 3.4 для приклада наведені деякі характеристики одного із серверів системи відеоспостереження за студентами під час тестування для контролю рівня знань (MCU Lucent Technologies), що визначають його сумісність із іншим устаткуванням.

Сервер системи відеоспостереження за студентами під час тестування для контролю рівня знань, підтримує відеостандарти, звукові стандарти й стандарти даних, перераховані в таблицях 3.5-3.6.

Функціональна схема системи відеоспостереження за студентами під час тестування для контролю рівня знань зображена на рисунку 3.2.

Термінали системи відеоспостереження за студентами під час тестування для контролю рівня знань

У системі системи відеоспостереження за студентами під час тестування для контролю рівня знань можуть бути використані моделі групових напольних,

компактних і настільних відеотерміналів різних виробників. Властивості систем різних фірм у кожній із груп досить близькі.

Ці засоби дозволяють працювати по протоколах H.320 і H.323 і поряд із груповими й компактними терміналами системи відеоспостереження за студентами під час тестування для контролю рівня знань можуть бути включені в корпоративну систему системи відеоспостереження за студентами під час тестування для контролю рівня знань.

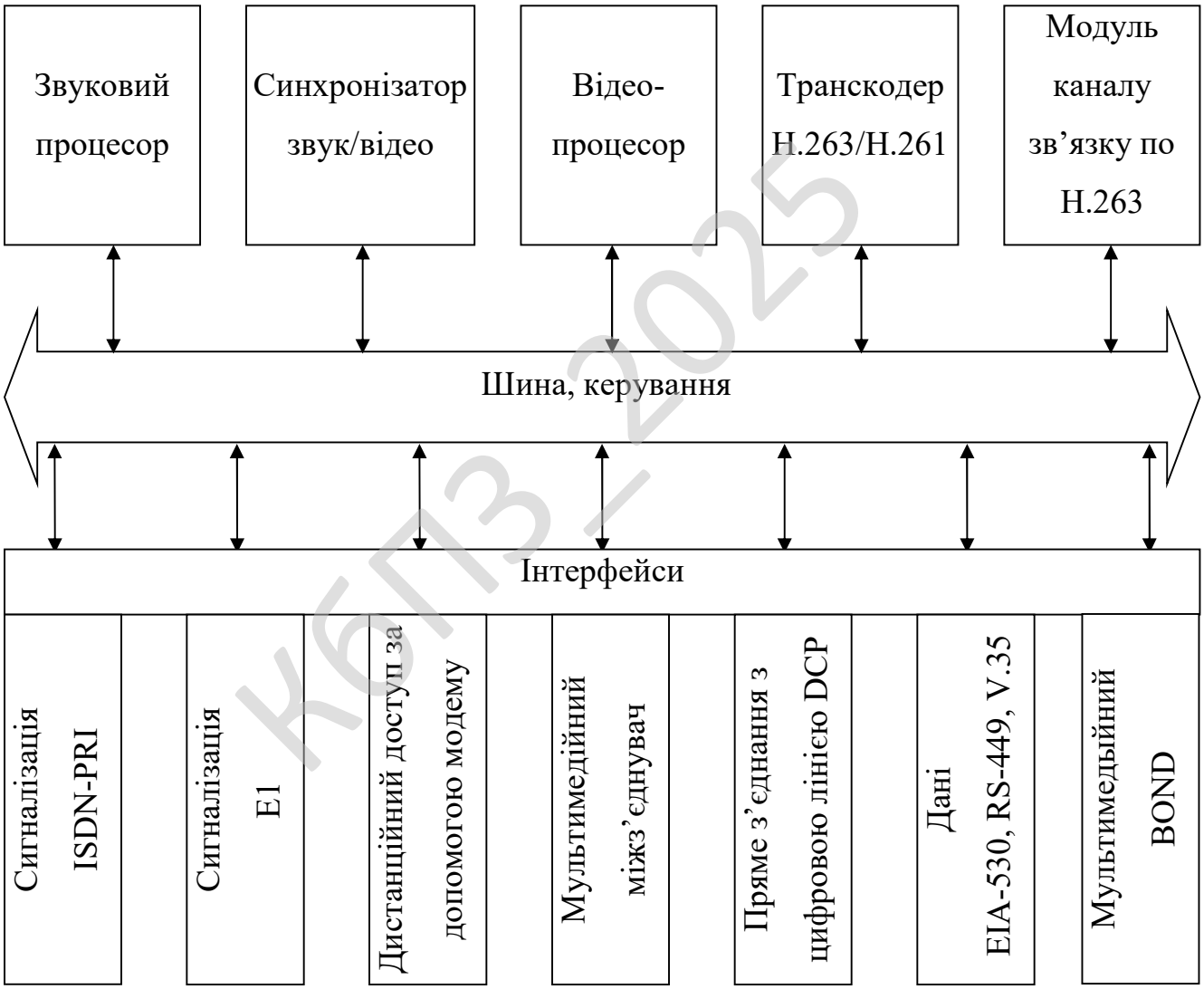


Рисунок 3.2 – Функціональна схема системи

Комунікаційні властивості встаткування системи відеоспостереження за студентами під час тестування для контролю рівня знань

Основою технологічного встаткування системам відеоспостереження за студентами під час тестування для контролю рівня знань, служать термінали різних конфігурацій і сервер системи відеоспостереження за студентами під час тестування для контролю рівня знань, із системою керування. Сервер системи відеоспостереження за студентами під час тестування для контролю рівня знань, є головним телекомунікаційним засобом багатоточечної системи відеоспостереження за студентами під час тестування для контролю рівня знань, і має продуктивні цифрові інтерфейси для високошвидкісної передачі голосових повідомлень, зображень і даних. Він містить інтерфейси цифрових мереж інтегрованих служб (ISDN): для базової цифрової швидкості (BRI) і первинної цифрової швидкості (PRI). Крім того, він підтримує протокол прямих цифрових з'єднань (DCP). При з'єднанні ISDN-PRI на рівні первинної групи групостворення E1 застосовується формат 30B + D, а при ISDN-BRI, – формат 2B + D, де B (64 кбіт/с) – потік для основної інформації служби, а D (16 кбіт/с) – потік керування й сигналізації для приєднаних каналів B. Протокол цифрових з'єднань DCP застосовується для зв'язку терміналів системи відеоспостереження за студентами під час тестування для контролю рівня знань, із системою керування за допомогою високошвидкісних і мультимедійних з'єднань (MML). Сервер системи відеоспостереження за студентами під час тестування для контролю рівня знань, може підключатися до кодеків H.320, H.CTX для передачі даних через інтерфейси EIA-530, RS449 або V.35 із застосуванням RS366 для сигналізації (дозвона). У випадку надання смуги на вимогу (BONDing) можна здійснювати реалізацію системи відеоспостереження за студентами під час тестування для контролю рівня знань, без використання каналів ISDN-PRI і широкополосних каналів типу H0.

Можливі схеми з'єднання терміналів системи відеоспостереження за студентами під час тестування для контролю рівня знань

Термінали системи відеоспостереження за студентами під час тестування для контролю рівня знань, з'єднуються один з одним за допомогою каналів зв'язку й коммунікаторів. Для багатоточечної системи відеоспостереження за студентами під час тестування для контролю рівня знань, головним коммунікатором є багатоточечний сервер системи відеоспостереження за студентами під час тестування для контролю рівня знань. Залежно від взаємного розташування терміналів і сервера системи відеоспостереження за студентами під час тестування для контролю рівня знань, режимів роботи пристроїв, а також використовуваних мереж зв'язки можливі різноманітні схеми з'єднання терміналів і серверів системи відеоспостереження за студентами під час тестування для контролю рівня знань. При значній взаємній далекості терміналів системи відеоспостереження за студентами під час тестування для контролю рівня знань, зв'язок між ними й сервером, а також між ними самими (в окремих випадках) здійснюється з використанням каналів магістральних, міжрегіональних і регіональних мереж зв'язку.

Таблиця 3.4 – Можливості транскодування

Категорія	Стандарти й параметри інтерфейсів
Транскодування сигналів зображення	G.711 і G.722
	G.711 і G.728
	G.711 і G.723 (через Gateway)
	Частота кадрів – від 7,5 до 30 кадр/с Розрішення – CIF/QCIF Стиск – H.261/H.263
Доступ до мережі	Цифрова швидкість – від 56 до 768 кбіт/с Режими – BONDing, багатошвидкісний, багатоканальний
Багатоточечні протоколи	H.320 і H.323
Конференція даних (T.120)	Допускається в комбінованих конференціях Змішання H.320/H.323 (через Gateway)

Таблиця 3.5 – Відеостандарти, підтримувані сервером системи відеоспостереження за студентами під час тестування для контролю рівня знань

Відеостандарт МСЕ-Т	Найменування, зміст
Н.221	Стандарт структури кадрів для системам відеоспостереження за студентами під час тестування для контролю рівня знань,
Н.230	Стандарт кадрової синхронізації для системам відеоспостереження за студентами під час тестування для контролю рівня знань,
Н.231	Стандарт системам відеоспостереження за студентами під час тестування для контролю рівня знань,, що визначає з'єднання між звуковізуальними терміналами
Н.242	Стандарт, що визначає системи для подання з'єднань між звуковізуальними терміналами
Н.243	Рекомендації ITU-T: процедура встановлення зв'язку між трьома або більше аудіовізуальними терміналами, що використовують канали зі швидкістю передачі цифрової інформації до 2 Мб/с.
Н.261	Стандарт відеокодека для звуковізуальних служб зі швидкістю Н.320
Н.263	Кодування й декодування зображення для передачі з низькою швидкістю з поліпшеними характеристиками і якістю по каналах Н.261

Таблиця 3.6 – Звукові стандарти, підтримувані сервером системи відеоспостереження за студентами під час тестування для контролю рівня знань

Звуковий стандарт МСЕ-Т	Параметри якості	Смуга, бітна швидкість
G.711	3,5 кГц	48/56/64 кбіт/с
G.722	7 кГц	48 кбіт/с на швидкості N*56 кбіт/с
		56 кбіт/с на швидкості N*64 кбіт/с
G.728	3.5 кГц	16 кбіт/с

Таблиця 3.7 – Стандарти даних, підтримувані сервером системи відеоспостереження за студентами під час тестування для контролю рівня знань

Стандарти МСЕ-Т	Найменування, зміст
Т. 122	Багатоточечна служба зв'язку для звукографії й системам відеоспостереження за студентами під час тестування для контролю рівня знань,
Т.123	Стеки протоколів ISDNProfile-MLP для застосувань звукографії й системам відеоспостереження за студентами під час тестування для контролю рівня знань,
Т.124	Специфікація GCC (Genetic Conference Protocol)
Т. 125	Протокол MC5
Т.126	Протокол для нерухливих зображень і описів при багатоточечному зв'язку
Т.127	Протокол перетворень багатоточечних бінарних файлів

У системі системи відеоспостереження за студентами під час тестування для контролю рівня знань зв’язок може вироблятися по протоколах Н.320 і Н.323 МСЕ-Т. Зв'язок терміналів системи відеоспостереження за студентами під час тестування для контролю рівня знань, що працюють по протоколі Н.323, із сервером системи відеоспостереження за студентами під час тестування для

контролю рівня знань, може здійснюватися через вузли (Gateway). З'єднання вузла Gateway із сервером системи відеоспостереження за студентами під час тестування для контролю рівня знань, виробляється по протоколах BRI і V.35/RS336.

Підключення сервера системи відеоспостереження за студентами під час тестування для контролю рівня знань, до магістральних, міжрегіональних і регіональних каналів зв'язку може вироблятися через маршрутизатор, магістральний мультиплексор і цифрову АТМ. З'єднання сервера із цими пристроями може здійснюватися з використанням протоколів PRI/E1, G.703. Підключення терміналів системи відеоспостереження за студентами під час тестування для контролю рівня знань, до магістральних, міжрегіональних і регіональних каналів зв'язку може відбуватися через мультиплексори земних станцій (HUB) і абонентських станцій (VSAT) супутникового зв'язку, магістральні мультиплексори (MUX) і АТМ. Залежно від відстані між терміналами й цим устаткуванням з'єднання можуть здійснюватися або безпосередньо з використанням протоколу V.35, або із застосуванням модемів або інверсних мультиплексорів.

Деякі особливості роботи підключення терміналів системи відеоспостереження за студентами під час тестування для контролю рівня знань, до каналоутворюючого устаткування

Безпосереднє підключення термінала системи відеоспостереження за студентами під час тестування для контролю рівня знань, до каналоутворюючому встаткування допускається в тих випадках, коли довжина сполучних кабелів невелика. У випадках, коли термінали й каналоутворююче встаткування вилучені друг від друга, для їхнього з'єднання повинні використовуватися модеми або інверсні мультиплексори.

Для входу в супутникову мережу термінали системи відеоспостереження за студентами під час тестування для контролю рівня знань, повинні бути підключені до мультиплексору центральної, вузлової або абонентської земної

станції (ЗС) з використанням інтерфейсної плати внутрішнього модуля ЗС. Залежно від відстані між терміналом системи відеоспостереження за студентами під час тестування для контролю рівня знань, і внутрішнім модулем земної станції підключення виробляється прямо або із застосуванням мультиплексора (модему). З'єднання рекомендується здійснювати з використанням протоколу V.35. Подібні способи підключення мають на увазі застосування твердої структури з'єднань, слабко пристосованої до подальшого розвитку. Вони доречні для використання в кінцеві (не вузлових) пунктах мережі системам відеоспостереження за студентами під час тестування для контролю рівня знань,.

У тих випадках, коли об'єкт системам відеоспостереження за студентами під час тестування для контролю рівня знань, розглядається як вузловий об'єкт, підключення термінала системи відеоспостереження за студентами під час тестування для контролю рівня знань, до внутрішнього модуля ЗС супутникового зв'язку варто здійснювати через мультиплексор.

Термінал системи відеоспостереження за студентами під час тестування для контролю рівня знань, підключається до інтерфейсного модуля мультиплексора прямо або через модем або інверсний мультиплексор, залежно від відстані між терміналом і мультиплексором.

На вузлових об'єктах з розвитою телефонною мережею доцільно підключати термінали до магістральних мереж через цифрову АТМ. При використанні цифровий АТМ у якості комунікатора потоків даних системам відеоспостереження за студентами під час тестування для контролю рівня знань, варто мати на увазі, що ці дані не повинні піддаватися стиску в АТМ, передбаченому для звичайних телефонних сигналів. Інакше кажучи, на період сеансу зв'язку повинна бути як би виділена смуга на вимогу для передачі цифрового потоку з необхідною бітною швидкістю (у загальному випадку 256 кбіт/с) для кожного з напрямків системам відеоспостереження за студентами під час тестування для контролю рівня знань.

На центральній ЗС або в центральному вузлі інформатизації з розгалуженими комунікаціями підключення термінала системи відеоспостереження за студентами під час тестування для контролю рівня знань, може бути здійснене через магістральний мультиплексор, цифрову телефонну станцію або маршрутизатор.

Підключення терміналів системи відеоспостереження за студентами під час тестування для контролю рівня знань, до магістральних, міжрегіональних і регіональних каналів зв'язку через мультиплексори, цифрові АТМ або маршрутизатори забезпечує ряд переваг у порівнянні із прямим підключенням до ЗС. По-перше, добре розвинені комунікаційні функції цих пристроїв дозволяють їх використовувати як вузли системам відеоспостереження за студентами під час тестування для контролю рівня знань, (хоча з деякими обмеженнями). По-друге, завдяки широкому діапазону інтерфейсів, властивим цим пристроям, досягається можливість зв'язку терміналів системи відеоспостереження за студентами під час тестування для контролю рівня знань, що діють у різнорідних мережах (наприклад, супутникових і наземних, телефонних і мультимедійних і т.п.). При цьому забезпечується гнучкість структури системи системам відеоспостереження за студентами під час тестування для контролю рівня знань, й подальше нарощування користувальницьких послуг, в основному програмними, а не апаратними засобами.

Робочі місця учасників процесу системи відеоспостереження за студентами під час тестування для контролю рівня знань

Загальні вимоги

Під «робочим місцем» розуміється комплект основного, додаткового й супутнього технологічного встаткування, а також інших засобів, розміщених на спеціально виділених площах і необхідних учасникові системи відеоспостереження за студентами під час тестування для контролю рівня знань, для виконання всіх передбачених технологією функцій і процесів. Робочі місця для різних учасників процесу відрізняються друг від друга по составі

встаткування і їхньому оформленню залежно від характеру технологічних процесів, виконуваних кожним з учасників. Робоче місце оснащується технологічним устаткуванням і технологічними меблями, засобами висвітлення, електропостачання й життєзабезпечення (опалення, вентиляції й кондиціювання повітря).

Загальні вимоги містять у собі також вимоги до технологічності, ергономіці, електропостачанню, висвітленню, шуму, безпеці, гігієні праці й профілактиці профзахворювань.

Робоче місце оператора сервера системи відеоспостереження за студентами під час тестування для контролю рівня знань

В состав устаткування оператора сервера системи відеоспостереження за студентами під час тестування для контролю рівня знань входить система резервування й керування відеоконференціями, професійний персональний комп'ютер з модемом і точка підключення до Інтернету.

Робочі місця користувачів системи відеоспостереження за студентами під час тестування для контролю рівня знань

Види робочих місць. Залежно від видів конференції, учасників і категорії термінального встаткування можуть бути виділені чотири основних види робітників місць.

Індивідуальне робоче місце абонента (користувача) системи відеоспостереження за студентами під час тестування для контролю рівня знань, збігається з його постійним робочим місцем в аудиторії. Групове робоче місце перебуває або в безпосередній близькості від постійного робочого місця одного з учасників системи відеоспостереження за студентами під час тестування для контролю рівня знань, або розміщається в окремо виділеному або пристосованому для цього приміщенні.

Робоче місце оператора системи відеоспостереження за студентами під час тестування для контролю рівня знань, територіально розташовується або поблизу групового робочого місця абонента системи відеоспостереження за

студентами під час тестування для контролю рівня знань, або на ділянці архіву системи відеоспостереження за студентами під час тестування для контролю рівня знань,, залежно від виконуваної роботи. До складу встаткування робочого місця оператора системи відеоспостереження за студентами під час тестування для контролю рівня знань, входить комплект устаткування монтажу, кодування відеофонограм і виготовлення архівних копій, а також термінал керування архівом системи відеоспостереження за студентами під час тестування для контролю рівня знань.

Робоче місце оператора сервера системи відеоспостереження за студентами під час тестування для контролю рівня знань, розташовується на ділянці зв'язку поблизу сервера системи відеоспостереження за студентами під час тестування для контролю рівня знань, і має термінал системи керування.

Робоче місце користувача системи відеоспостереження за студентами під час тестування для контролю рівня знань включає абонентський термінал системи відеоспостереження за студентами під час тестування для контролю рівня знань, індивідуального або групового застосування з відеокамерою, мікрофоном, відеотерміналом, гучномовцями й електронними блоками; додаткове встаткування (документальну відеокамеру, комп'ютер, додаткові відеокамеру, мікрофон і відеомагнітофон) і супутнє встаткування (відеопроектор, системи спецосвітлення й звукопідсилення). Залежно від призначення й цілей застосування робочого місця користувача додаткове й супутнє встаткування використовують у різній комплектації..

Вимоги до розміщення встаткування й людей

Розміщення встаткування й людей – користувачів системи відеоспостереження за студентами під час тестування для контролю рівня знань – повинне відповідати ряду суперечливих вимог, частина з яких нормована. По-перше, повинні бути виконані норми МСЕ-Р (сектора Р Міжнародного союзу електрозв'язку) і ЄСВ (Європейського союзу віщання), пропоновані до умов перегляду відеоматеріалу й прослуховуванню звукового матеріалу, по-друге, –

технічні вимоги розміщення кожного з видів устаткування, по-третє, враховані норми СНіП у частині електро- і пожаропожежебезпеки. Параметри приміщення й умови розміщення встаткування й людей підлягають розрахунку з урахуванням наступних норм і документів:

– співвідношення сторін приміщення – відповідно до «благозвучного» співвідношеннями Болта [1, 2];

– обсяг приміщення й час реверберації – відповідно до норм МСЕ-Р (МККР) серії BS [3, 4] і вимогами ЄСВ [5];

– мінімальна відстань гучномовців термінала системи відеоспостереження за студентами під час тестування для контролю рівня знань, від задньої стіни й розташування людей щодо гучномовців – відповідно до вимог МЕК [6] і МСЕ-Р [3, 7];

– шумові характеристики приміщення – відповідно до норм МСЕ-Р і ЄСВ [2, 3, 7];

– розташування людей щодо екрана відеомонітора – відповідно до вимог рекомендацій МСЕ-Р серії ВТ [8 -10];

– розміщення людей щодо стін приміщення – відповідно до вимог СНіП [11-13], а також з урахуванням вимог, пропонованих до висвітлення тла при відеозйомці.

Вимоги до розмірів приміщення визначаються з урахуванням акустичних вимог до часу реверберації, частотній характеристиці й раннім відбиттям [5, 7].

Оптимальне планування робочого місця досягається на підставі інженерного розрахунку.

Особливу увагу варто приділяти акустиці приміщень. Акустична обробка поверхонь стін, стелі й підлоги повинна бути такий, щоб запобігти раннім відбиттям звуку, видаваного учасниками й вихідного від гучномовців.

Для проведення групових системи відеоспостереження за студентами під час тестування для контролю рівня знань важливо ретельно проробити

світлотехнічні рішення. Так, колірна температура штучних джерел світла в залі повинна бути однакової щоб уникнути порушення передачі кольору.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. Після початку роботи розробленого ПЗ ми потрапляємо до головного блоку системи звідки через ланку дій відбувається наступне:

– Інтерфейс ПЗ.

– Налаштування системи.

– Налаштування параметрів ПЗ.

– Налаштування параметрів відео.

– Обробник помилок.

– З’єднання з мережею.

– Обробка даних Web-камери.

– Відео-пошта.

– Запис відео-розмов.

– Відеочат.

– Встановлення профілю користувача.

– Статус.

– Індикатор стану.

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

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.

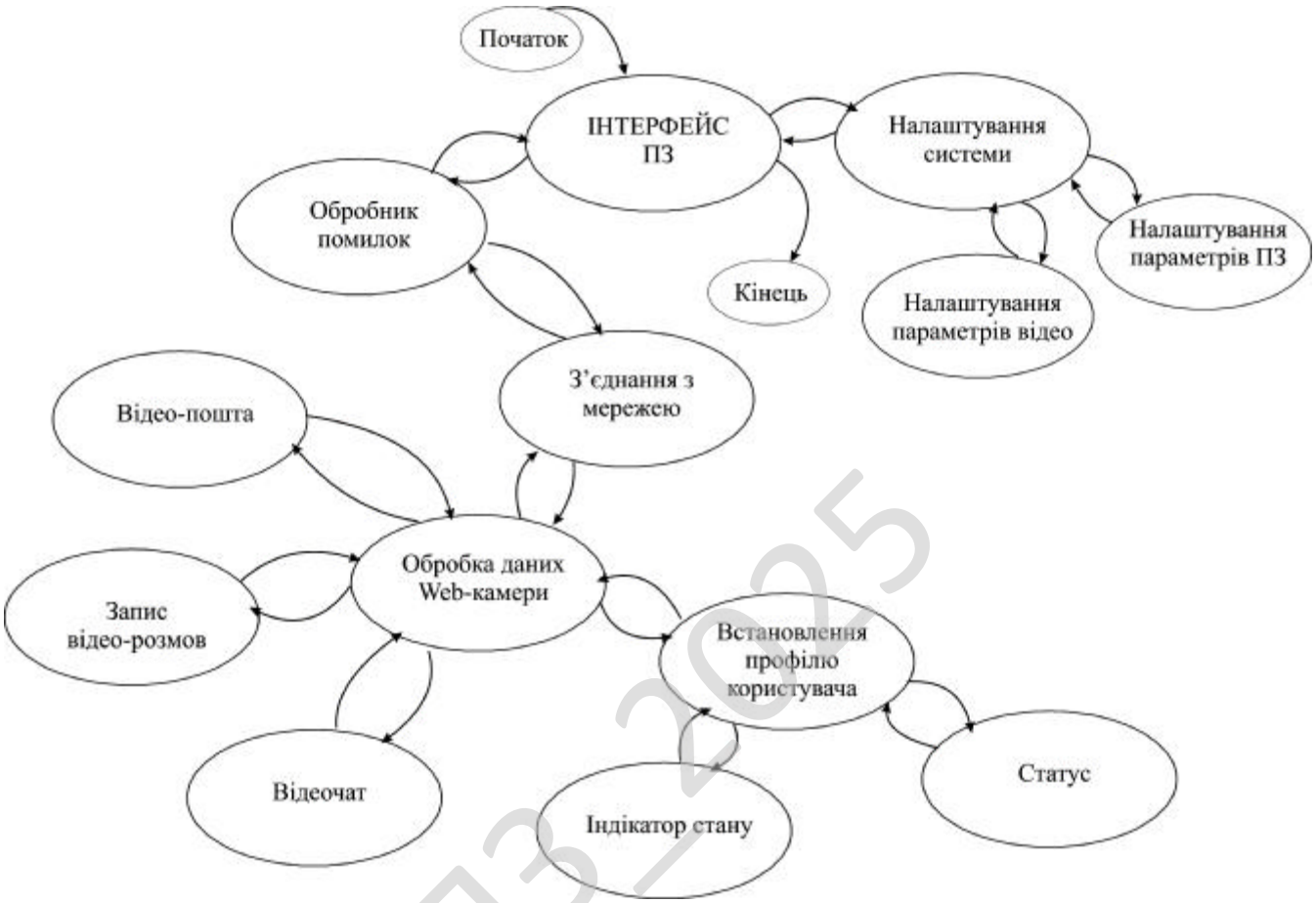


Рисунок 3.3 – Діаграма взаємодії процесів

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

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З якої 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 – перевірка поточного стану з завершенням роботи розробленого ПЗ.

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.

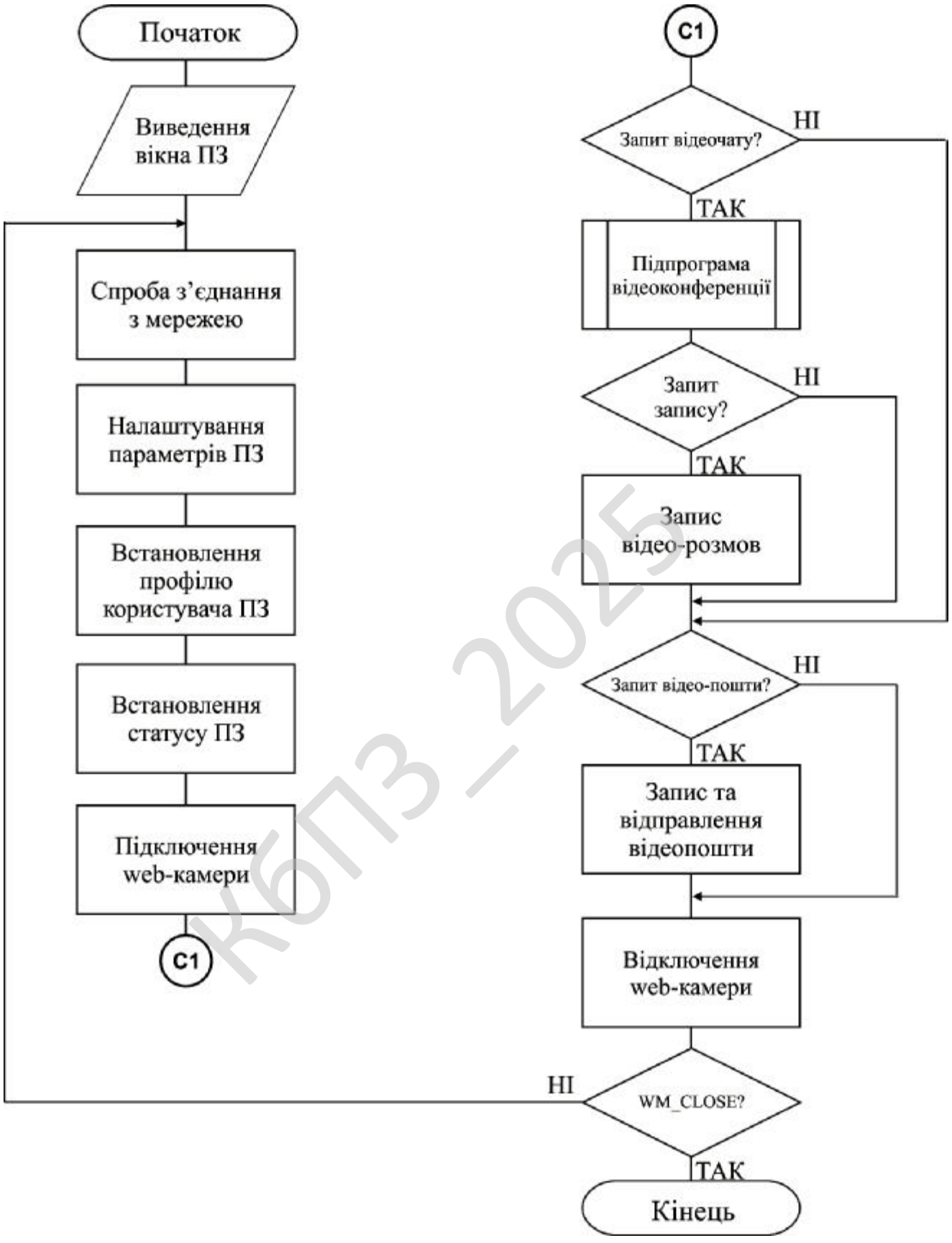


Рисунок 4.1 – Блок схема основної програми

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою. Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки.

Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

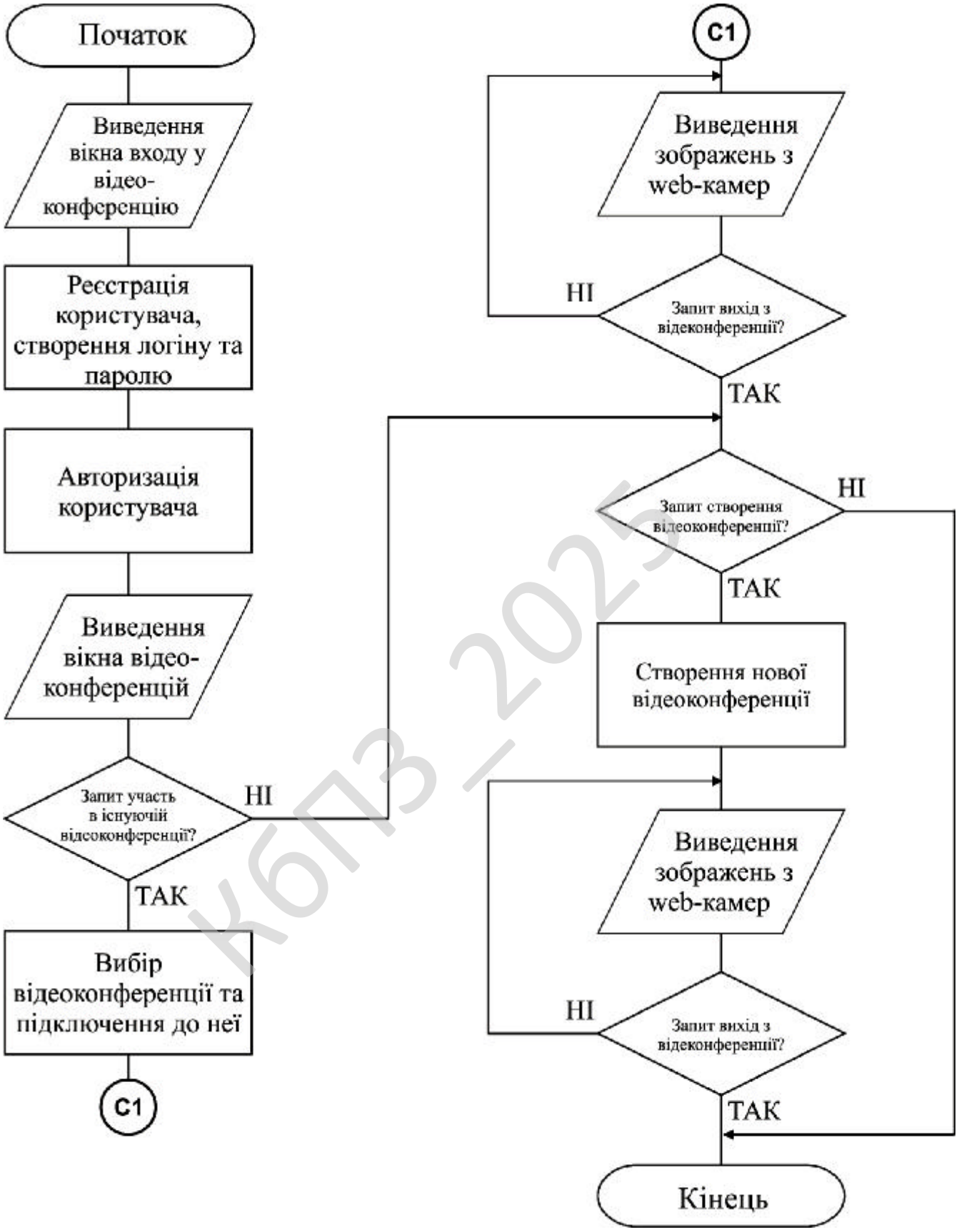


Рисунок 4.2 – Блок схема підпрограми

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності.

Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем я враховував, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю системи відеоспостереження за студентами під час тестування для контролю рівня знань.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою. Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

Опис системи

Система відеоспостереження за студентами під час тестування для контролю рівня знань складається з декількох основних компонентів.

Вона забезпечує надійний моніторинг та виявлення порушень під час проведення іспитів.

Основними елементами системи є камера спостереження, сервер обробки відео, програмне забезпечення для розпізнавання обличчя та виявлення шахрайства, база даних для збереження інформації та клієнтський інтерфейс для перегляду результатів.

Система використовує мову програмування Python. Основні модулі, що застосовуються у розробці, включають OpenCV для обробки відео, TensorFlow для машинного навчання, SQLite для роботи з базою даних та Flask для реалізації веб-інтерфейсу.

Принцип роботи системи базується на потоковому відеозаписі з камер, що передають зображення на сервер.

Сервер обробляє відеопотік, застосовуючи алгоритми розпізнавання облич та відстеження поведінки студентів. Виявлення порушень відбувається шляхом аналізу рухів голови, зміни фокусування погляду, присутності сторонніх об'єктів та використання заборонених пристроїв.

Основні функції системи включають:

– Захоплення відео з камер.

– Розпізнавання облич студентів та ідентифікація особи.

– Виявлення порушень у поведінці.

– Збереження відеофрагментів у разі виявлення підозрілої активності.

– Формування звіту для викладача.

Архітектура системи має серверно-клієнтську модель. Сервер обробляє відео, аналізує дані та зберігає їх у базі даних. Клієнтська частина забезпечує доступ до аналітики через веб-інтерфейс.

Обчислення продуктивності системи базуються на аналізі швидкості обробки відео. Використання бібліотеки OpenCV дозволяє обробляти відеопотік зі швидкістю до 30 кадрів за секунду.

Система використовує нейромережі для аналізу зображень, що забезпечує точність розпізнавання на рівні 95 відсотків. Оптимізація алгоритмів дозволяє знизити затримку обробки до 100 мілісекунд на один кадр.

Загальний розрахунок необхідної потужності обчислювальних ресурсів виконується з урахуванням кількості студентів, що тестуються одночасно.

Якщо одночасно тестування проходять 50 студентів, кожен з яких знаходиться під контролем окремої камери, то загальне навантаження на сервер складає 50 відеопотоків.

В середньому, система використовує 2 ГБ оперативної пам'яті на один відеопотік.

Таким чином, загальний обсяг необхідної пам'яті становить 100 ГБ. Процесорний час розподіляється залежно від моделі CPU та можливості розпаралелювання задач.

Система відеоспостереження дозволяє автоматизувати контроль за студентами під час тестування, що підвищує об'єктивність оцінювання та зменшує ймовірність порушень.

```
import cv2
import face_recognition
import numpy as np
import sqlite3
import time
from flask import Flask, render_template, Response
app = Flask(__name__)

# Підключення до бази даних
conn = sqlite3.connect("students.db", check_same_thread=False)
cursor = conn.cursor()
cursor.execute('''CREATE TABLE IF NOT EXISTS violations
                (id INTEGER PRIMARY KEY AUTOINCREMENT, student_name TEXT,
                 timestamp TEXT, violation_type TEXT)''')
conn.commit()

# Ініціалізація камери
video_capture = cv2.VideoCapture(0)

# Завантаження зразків облич студентів
known_face_encodings = []
known_face_names = []

students = {"John Doe": "john.jpg", "Jane Smith": "jane.jpg"}

for name, filename in students.items():
    image = face_recognition.load_image_file(filename)
    encoding = face_recognition.face_encodings(image)[0]
    known_face_encodings.append(encoding)
    known_face_names.append(name)

def detect_violations(frame, face_names):
    violations = []
```

```
            for name in face_names:
                if name == "Unknown":
                  violations.append(("Невідомий студент",
                  time.strftime("%Y-%m-%d %H:%M:%S"), "Незареєстроване обличчя"))
               if len(set(face_names)) > 1:
                  violations.append((name,    time.strftime("%Y-%m-%d    %H:%M:%S"),
                   "Більше однієї особи в кадрі"))
            return violations

        def save_violations(violations):
            for violation in violations:
                cursor.execute("INSERT INTO violations (student_name, timestamp,
                 violation_type) VALUES (?, ?, ?)", violation)
            conn.commit()

        def generate_frames():
            while True:
                ret, frame = video_capture.read()
                if not ret:
                    break

                rgb_frame = frame[:, :, ::-1]
                face_locations = face_recognition.face_locations(rgb_frame)
                face_encodings = face_recognition.face_encodings(rgb_frame,
face_locations)

                face_names = []
                for face_encoding in face_encodings:
                    matches = face_recognition.compare_faces(known_face_encodings,
                              face_encoding)
                    name = "Unknown"

                    face_distances =
face_recognition.face_distance(known_face_encodings, face_encoding)
                    best_match_index = np.argmin(face_distances)
                    if matches[best_match_index]:
                        name = known_face_names[best_match_index]
                    face_names.append(name)
                for (top, right, bottom, left), name in zip(face_locations,
                     face_names):
                    color = (0, 255, 0) if name != "Unknown" else (0, 0, 255)
                    cv2.rectangle(frame, (left, top), (right, bottom), color, 2)
```

```
            cv2.putText(frame, name, (left + 6, bottom - 6),
            cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 1)
            violations = detect_violations(frame, face_names)
            save_violations(violations)
            ret, buffer = cv2.imencode('.jpg', frame)
            frame = buffer.tobytes()
            yield (b'--frame\r\n'
                   b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
        @app.route('/')
def index():
    return render_template('index.html')

@app.route('/video_feed')
def video_feed():
return Response(generate_frames(),
        mimetype='multipart/x-mixed-replace; boundary=frame')
if __name__ == "__main__":
    app.run(debug=True)
video_capture.release()
cv2.destroyAllWindows()
```

Цей частина коду спланована на реалізацію функцій:

– Захоплення відео з камери.

– Розпізнавання облич студентів.

– Виявлення підозрілих дій, таких як присутність сторонніх осіб.

– Збереження порушень у базу даних.

– Відображення відеопотоку через веб-інтерфейс.

Веб-інтерфейс реалізовано за допомогою Flask. Він дозволяє переглядати відеопотік у реальному часі. Виявлені порушення автоматично записуються у базу даних.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм SHACAL-1, який заснований на перетвореннях алгоритму SHA-1. SHACAL-1 шифрує 160-бітний блок даних з використанням

512-бітного ключа шифрування. Допускається використання більше коротких ключів шифрування (не коротше 128 біт), які перед виконанням розширення ключа (процедура розширення ключа також успадкована від SHA-1 і буде описана нижче) повинні бути доповнені нульовими бітами для досягнення 512-бітного розміру.

В алгоритмі SHACAL-1 передбачено 80 раундів шифрування. Шифруєме повідомлення представляється у вигляді п'яти 32-бітних субблоків A, B, C, D і E, над якими в кожному раунді виконуються наступні дії:

$$A_{i+1} = K_i + (A_i <<< 5) + f_i(B_i, C_i, D_i) + E_i + M_i,$$

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

$$C_{i+1} = B_i <<< 30,$$

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

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

де i – номер раунду (i = 0...79),

K_i – фрагмент розширеного ключа для і-го раунду,

f_i – функція для і-го раунду (див. нижче),

<<< – операція побітового циклічного зрушення вліво,

M_i – константи, що модифікують, певні в такий спосіб:

Раунди	Значення константи
0...19	5A827999
20...39	6ED9EBA1
40...59	8F1BBCDC
60...79	CA62C1D6

Використовувані в раундах функції f_i визначені так:

Раунди	Функція
0...19	$f(x,y,z)=(x\&y)\|(x'\&z)$
20...39,60…79	$f(x,y,z)=x \oplus y \oplus z$
40...59	$f(x,y,z)=(x \oplus y)\|(x \oplus z)\|(y \oplus z)$

У таблиці символами &, | і ⊕ позначені, відповідно, побітові логічні операції «і», «або» й «або, що виключає» (XOR); *x*' позначає побітовий комплемент до *x*.

Шифртекстом є конкатенація вмісту змінних A_{80}, B_{80}, C_{80}, D_{80} і E_{80}.

Процедура розширення ключа в алгоритмі SHACAL-1 також досить проста, вона виконується у два етапи:

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 $K_{16} \ldots K_{79}$ обчислюються з перших 16 фрагментів у такий спосіб:

$$K_i = (K_{i-3} \oplus K_{i-8} \oplus K_{i-14} \oplus K_{i-16}) <<< 1.$$

Раунди розшифрування виконуються у зворотній послідовності; кожний з них має на увазі виконання наступних операцій:

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

$$B_i = C_{i+1} <<< 2,$$

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

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

$$E_i = K'_i + (B_{i+1} <<< 5)' + f_i'(C_{i+1} <<< 2, D_{i+1}, E_{i+1}) + A_{i+1} + M'_i + 4.$$

Тут запис $f'(x)$ позначає побітовий комплемент результату виконання операції $f(x)$.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ яке зображено на рисунку 5.1, а. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні розділи:

– Меню: Дії; Сервіс; Довідка.

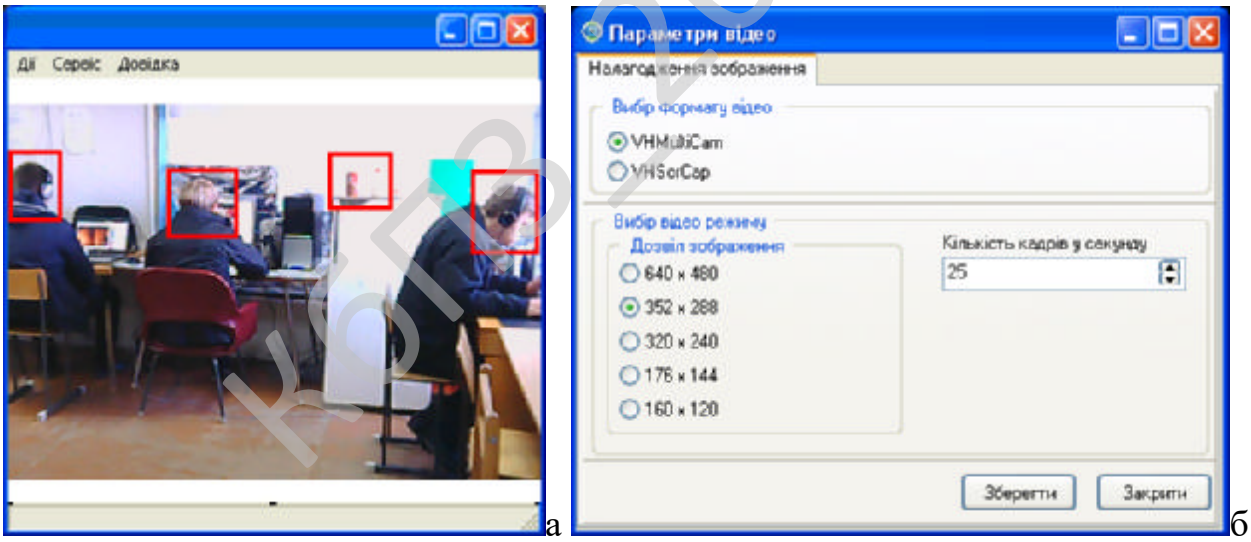
– Вікно виведення відеоданих з розпізнаванням руху.

На рисунку 5.1, б зображено вікно параметрів програми, з нього ми бачимо, що у програмі можливо встановити наступні параметри:

– Вибір формату відео.

– Вибір розрішення зображення.

– Вибір кількості кадрів у секунду.



а б

Рисунок 5.1 – Вікна розробленого 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.

Обрано умови розповсюдження – Shareware. Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання. Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму. В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

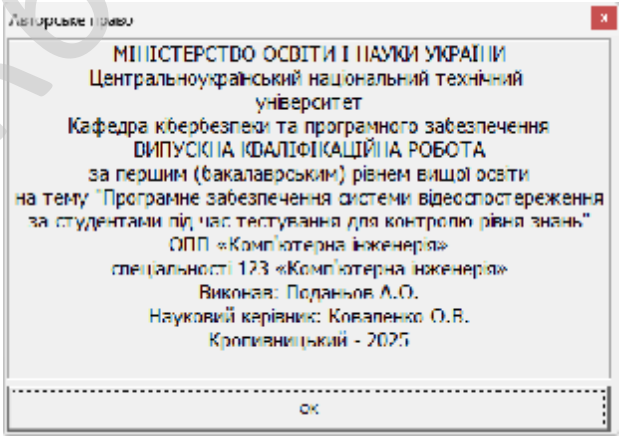


Рисунок 5.2 – Авторське право

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи відеоспостереження за студентами під час тестування для контролю рівня знань.

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 студентами під час тестування для контролю рівня знань.

– Досліджена система відеоспостереження за студентами під час тестування для контролю рівня знань.

– На основі отриманих результатів досліджень створена програмна реалізація системи відеоспостереження за студентами під час тестування для контролю рівня знань.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання відеоспостереження за студентами під час тестування для контролю рівня знань.

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 студентами під час тестування для контролю рівня знань. 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-1.

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. Henry Lloyd. Interactive Computer Graphics. States Academic Press. 2022. 247 p.

2. Ranjan Parekh. Fundamentals of Image, Audio, and Video Processing Using MATLAB®With Applications to Pattern Recognition. CRC Press. 2021. 406 p.

3. Alasdair McAndrew. A Computational Introduction to Digital Image Processing. Chapman & Hall. 2021. 560 p.

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

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

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

7. 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.

8. 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.

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

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

11. 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.

12. 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

13. 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.

14. 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.

15. 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.

16. 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,

17. 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.

18. 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

19. 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.

20. 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.

21. 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.

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

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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.

44. 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.

45. 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.

46. 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.

47. 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.

48. 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.

49. 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.

50. 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).

51. 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

52. 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.

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

54. 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.

55. 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.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи відеоспостереження за студентами під час тестування для контролю рівня знань.

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 студентами під час тестування для контролю рівня знань.

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 студентами під час тестування для контролю рівня знань;

– 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 – 62 аркуші.

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

_____ Коваленко О.В.

Програмне забезпечення системи відеоспостереження за студентами під час тестування для контролю рівня знань

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 17

Літера: РП

Кропивницький – 2025 року

Основна програма

```
#Import necessary libraries
import cv2
import numpy as np
import sqlite3
import datetime
import threading
import time
import os
import shutil
import random

#Initialize the SQLite database for logging surveillance events
def init_database():
    #Connect to the SQLite database file
    conn = sqlite3.connect('surveillance_events.db')
    #Create table for events if it does not exist already
    conn.execute('''
    CREATE TABLE IF NOT EXISTS events (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    timestamp TEXT,
    event_type TEXT,
    details TEXT
    )
    ''')
    #Commit changes to the database
    conn.commit()
    #Return the database connection object
    return conn

#Log an event into the database with event type and details
def log_event(conn, event_type, details):
    #Obtain current timestamp in the desired format
    timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    #Insert the event record into the events table
    conn.execute("INSERT INTO events (timestamp, event_type, details) VALUES (?,
?, ?)", (timestamp, event_type, details))
    #Commit the insert operation to the database
    conn.commit()
    #If the event is suspicious, simulate sending an alert
    if event_type in ["No Face Detected", "Multiple Faces Detected"]:
        simulate_alert_system(event_type, details)

#Retrieve all events from the database for analysis
def get_all_events(conn):
    #Execute SQL query to select all event records
    cursor = conn.execute("SELECT * FROM events")
    #Fetch all rows from the executed query
    events = cursor.fetchall()
    #Return the list of events
    return events

#Calculate event statistics from the database records
def get_event_statistics(conn):
    #Execute query to count events grouped by event type
    cursor = conn.execute("SELECT event_type, COUNT(*) FROM events GROUP BY
event_type")
    #Fetch all statistic rows
    stats = cursor.fetchall()
    #Return event statistics as a list of tuples
    return stats

#Print event statistics on the console
def print_event_statistics(conn):
    #Retrieve event statistics using get_event_statistics function
    stats = get_event_statistics(conn)
```

```
    #Print header for event statistics
    print("Event Statistics:")
    #Iterate over each statistic tuple and print details
    for stat in stats:
        print(f"Event Type: {stat[0]}, Count: {stat[1]}")

#Simulate sending an alert for suspicious surveillance events
def simulate_alert_system(event_type, details):
    #Print an alert message to the console
    print(f"ALERT: {event_type} - {details}")

#Simulate system load check to monitor performance metrics
def simulate_system_load():
    #Print a message indicating system load check
    print("Checking system load...")
    #Simulate delay in checking system load
    time.sleep(0.2)
    #Generate a random system load percentage
    load = random.uniform(0, 100)
    #Print the simulated system load value
    print(f"System load at {load:.2f}%")

#Simulate memory usage check to monitor resource consumption
def simulate_memory_usage():
    #Print a message indicating memory usage measurement
    print("Measuring memory usage...")
    #Simulate delay in measuring memory usage
    time.sleep(0.2)
    #Generate a random memory usage value in MB
    usage = random.uniform(100, 4096)
    #Print the simulated memory usage
    print(f"Memory usage: {usage:.2f} MB")

#Backup the current surveillance events database to a new file
def backup_database(conn):
    #Generate a backup filename using the current timestamp
    backup_filename =
f"backup_{datetime.datetime.now().strftime('%Y%m%d_%H%M%S')}.db"
    try:
        #Execute a VACUUM command to optimize the database
        conn.execute("VACUUM")
        #Copy the existing database file to the backup filename
        shutil.copy('surveillance_events.db', backup_filename)
        #Print a message indicating successful backup creation
        print(f"Database backup created: {backup_filename}")
    except Exception as e:
        #Print an error message if backup fails
        print(f"Database backup failed: {str(e)}")

#Simulate analysis of surveillance data to generate academic integrity reports
def simulate_data_analysis(conn):
    #Print a message indicating start of data analysis
    print("Analyzing surveillance data for academic integrity...")
    #Simulate delay for data analysis
    time.sleep(1)
    #Retrieve event statistics from the database
    stats = get_event_statistics(conn)
    #Print header for analysis results
    print("Data analysis complete. Results:")
    #Iterate over each event statistic and print its result
    for event_type, count in stats:
        print(f"{event_type}: {count}")


#Simulate monitoring environmental conditions such as lighting and noise levels
def monitor_environmental_conditions():
    #Print a message indicating environmental conditions are being monitored
    print("Monitoring environmental conditions...")
```

```
    #Simulate delay for environmental check
    time.sleep(0.2)
    #Print a message indicating optimal environmental conditions
    print("Environmental conditions are optimal for surveillance.")

#Simulate camera adjustments during the surveillance session
def simulate_camera_adjustment():
    #Print a message indicating camera adjustment process
    print("Adjusting camera settings...")
    #Simulate delay for camera adjustments
    time.sleep(0.2)
    #Print a message indicating successful camera adjustment
    print("Camera adjusted successfully.")

#Perform system diagnostics to check all components of the surveillance system
def perform_system_diagnostics():
    #Print a message indicating system diagnostics are starting
    print("Performing system diagnostics...")
    #Simulate delay for diagnostics
    time.sleep(0.3)
    #Print a message indicating diagnostics have completed
    print("Diagnostics completed. All systems functional.")

#Define the SurveillanceSystem class that encapsulates video surveillance
functionality
class SurveillanceSystem:
    #Constructor for the SurveillanceSystem class
    def __init__(self, video_source=0):
        #Store the video source index (default webcam is 0)
        self.video_source = video_source
        #Open the video capture device using OpenCV
        self.cap = cv2.VideoCapture(self.video_source)
        #Load the Haar cascade classifier for face detection from OpenCV data
        self.face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
        #Initialize the database connection by calling init_database function
        self.conn = init_database()
        #Flag to indicate whether the surveillance system is running
        self.running = False
        #Counter to generate unique filenames for saved snapshots
        self.snapshot_counter = 0
        #Lock for thread-safe operations during database access
        self.lock = threading.Lock()
        #Flag to indicate active test session
        self.test_session_active = False

    #Method to start the surveillance system and initiate video capture
processing
    def start(self):
        #Set the running flag to True to indicate surveillance is active
        self.running = True
        #Set the test session active flag to True
        self.test_session_active = True
        #Start a new thread to run the surveillance loop concurrently
        self.thread = threading.Thread(target=self.run_surveillance)
        #Start the surveillance thread
        self.thread.start()

    #Method to stop the surveillance system and release resources
    def stop(self):
        #Set the running flag to False to signal termination of the surveillance
loop
        self.running = False
        #Wait for the surveillance thread to finish execution
        self.thread.join()

        #Release the video capture resource
        self.cap.release()
```

```
        #Destroy all OpenCV windows that were opened
        cv2.destroyAllWindows()
        #Set the test session active flag to False
        self.test_session_active = False

    #Main surveillance loop that continuously captures and processes video
frames
    def run_surveillance(self):
        #Continue looping while the running flag is True
        while self.running:
            #Read a frame from the video capture device
            ret, frame = self.cap.read()
            #If frame capture is unsuccessful, continue to the next iteration
            if not ret:
                continue
            #Process the captured frame for face detection and event logging
            self.process_frame(frame)
            #Simulate a short delay to control frame processing rate
            time.sleep(0.05)

    #Process each frame to detect faces and log corresponding events
    def process_frame(self, frame):

        #Convert the captured frame to grayscale for easier processing
        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        #Detect faces in the grayscale frame using the Haar cascade classifier
        faces = self.face_cascade.detectMultiScale(gray_frame, scaleFactor=1.3,
minNeighbors=5)
        #If no faces are detected, log the event and save a snapshot
        if len(faces) == 0:
            log_event(self.conn, "No Face Detected", "No face found in the
current frame.")
            self.save_snapshot(frame, "noface")

        #If more than one face is detected, log the event as suspicious and save
a snapshot
        elif len(faces) > 1:
            log_event(self.conn, "Multiple Faces Detected", "Multiple faces
detected in the current frame.")
            self.save_snapshot(frame, "multiface")

        #If exactly one face is detected, log the event as normal behavior
        else:
            log_event(self.conn, "Face Detected", "A single face detected in the
current frame.")
        #Draw rectangles around each detected face for visual indication
        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)

        #Display the processed frame in a window titled 'Student Surveillance'
        cv2.imshow('Student Surveillance', frame)
        #Check if the 'q' key has been pressed to exit the surveillance loop
        if cv2.waitKey(1) & 0xFF == ord('q'):
            self.stop()

    #Save a snapshot image of the current frame when an event occurs
    def save_snapshot(self, frame, event_label):
        #Construct a unique filename for the snapshot using event label and
snapshot counter
        filename = f"snapshot_{event_label}_{self.snapshot_counter}.jpg"

        #Write the image frame to the file using OpenCV imwrite function
        cv2.imwrite(filename, frame)

        #Increment the snapshot counter for future snapshots
        self.snapshot_counter += 1
```

```
            #Log the snapshot saving event in the database
            log_event(self.conn, "Snapshot Saved", f"Snapshot saved as {filename}")

    #Query detailed event logs related to the current test session from the
database
    def query_test_session_details(self):
        #Define the SQL query to select all event records
        query = "SELECT * FROM events"
        #Acquire the lock for thread-safe database access
        with self.lock:
            cursor = self.conn.execute(query)
            #Fetch all event records from the query result
            rows = cursor.fetchall()
        #Return the list of event records
        return rows

    #Print all test session event details to the console for debugging purposes
    def print_test_session_details(self):
        #Retrieve event details by calling query_test_session_details method
        details = self.query_test_session_details()
        #Iterate over each event record and print it
        for event in details:
            print(event)

#Simulate the student login process and return a dummy student ID
def student_login():
    #Print a message indicating student login process has started
    print("Processing student login...")
    #Simulate delay for login processing
    time.sleep(0.5)
    #Return a dummy student identifier after login
    return "student_1234"

#Simulate the start of a test session and return a success flag
def start_test_session():
    #Print a message indicating that the test session is about to start
    print("Test session is starting...")
    #Simulate a delay before the session officially starts
    time.sleep(0.5)
    #Return True to indicate successful start of the test session
    return True

#Simulate the end of a test session and return a success flag
def end_test_session():
    #Print a message indicating that the test session is ending
    print("Test session is ending...")
    #Simulate a delay during session termination
    time.sleep(0.5)
    #Return True to indicate successful end of the test session
    return True

#Simulate additional background operations during the test session
def background_operations():
    #Monitor environmental conditions during the test session
    monitor_environmental_conditions()
    #Simulate camera adjustments if necessary
    simulate_camera_adjustment()
    #Perform system load check to ensure performance
    simulate_system_load()
    #Perform memory usage check to monitor resource consumption
    simulate_memory_usage()
    #Perform system diagnostics to check all components
    perform_system_diagnostics()

#Simulate periodic backup of the surveillance events database
def periodic_database_backup(conn, interval_seconds, stop_event):
```

```
    #Loop until stop_event is set to True
    while not stop_event.is_set():
        #Wait for the specified backup interval
        time.sleep(interval_seconds)
        #Perform database backup
        backup_database(conn)

#Main function to execute the complete surveillance system during a test session
def main():
    #Simulate student login process and obtain student ID
    student_id = student_login()
    #Print confirmation of successful student login
    print(f"Student {student_id} logged in successfully.")
    #Start the test session and verify successful initiation
    session_started = start_test_session()
    #If test session started successfully, proceed with surveillance
    if session_started:
        #Create an instance of the SurveillanceSystem for video monitoring
        surveillance_system = SurveillanceSystem(video_source=0)
        #Start the surveillance system to begin video capture and processing
        surveillance_system.start()
        #Set up a stop event for periodic database backup thread
        backup_stop_event = threading.Event()
        #Start the periodic database backup in a separate thread (interval set
to 30 seconds)
        backup_thread = threading.Thread(target=periodic_database_backup,
args=(surveillance_system.conn, 30, backup_stop_event))
        backup_thread.start()
        #Define the duration of the test session in seconds (simulation
duration)
        test_duration = 60
        #Record the start time of the test session
        start_time = time.time()
        #Loop until the defined test duration has elapsed
        while time.time() - start_time < test_duration:
            #Simulate background operations during the test session
            background_operations()

            #Pause for 1 second between background operations
            time.sleep(1)
            #Periodically print the test session event details for monitoring
            surveillance_system.print_test_session_details()

        #After the test duration, stop the surveillance system
        surveillance_system.stop()

       #Signal the backup thread to stop and wait for it to finish
        backup_stop_event.set()
        backup_thread.join()
    #End the test session and check for successful termination
    session_ended = end_test_session()
    #If the test session ended successfully, print a confirmation message
    if session_ended:
        print("Test session completed successfully.")
    else:
        print("Test session encountered an error during termination.")
    #Perform a final data analysis of the surveillance events
    simulate_data_analysis(surveillance_system.conn)
    #Print final event statistics from the database
    print_event_statistics(surveillance_system.conn)
    #Close the database connection gracefully
    surveillance_system.conn.close()

#Entry point of the program
if __name__ == '__main__':
    #Call the main function to start the surveillance system program
    main()
```

Файл VoiceAnalyzer.py

```
import cv2
import numpy as np
import time
import threading
import socket
import json
import matplotlib.pyplot as plt
import speech_recognition as sr
import csv
import random
from keras.models import load_model

class EmotionRecognition:
    def __init__(self, model_path):
        self.model = load_model(model_path)
        self.emotions = ["angry", "disgust", "fear", "happy", "sad", "surprise",
"neutral"]
    def preprocess(self, face_img):
        img = cv2.resize(face_img, (48, 48))
        img = img.astype("float32") / 255.0
        img = np.expand_dims(img, axis=0)
        img = np.expand_dims(img, axis=-1)
        return img
    def predict_emotion(self, face_img):
        preprocessed = self.preprocess(face_img)
        preds = self.model.predict(preprocessed)
        emotion_index = np.argmax(preds)
        return self.emotions[emotion_index]
    def analyze_frame(self, frame):
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        face_detector = cv2.CascadeClassifier(cv2.data.haarcascades +
"haarcascade_frontalface_default.xml")
        faces = face_detector.detectMultiScale(gray, 1.3, 5)
        results = []
        for (x, y, w, h) in faces:
            face_img = gray[y:y+h, x:x+w]
            emotion = self.predict_emotion(face_img)
            results.append({"position": (x, y, w, h), "emotion": emotion})
        return results

class VoiceAnalyzer:
    def __init__(self):
        self.recognizer = sr.Recognizer()
        self.microphone = sr.Microphone()
    def record_audio(self, duration=5):
        with self.microphone as source:
            audio = self.recognizer.record(source, duration=duration)
        return audio
    def analyze_audio(self, audio):
        try:
```

```
            text = self.recognizer.recognize_google(audio, language="en-US")
        except sr.UnknownValueError:
            text = ""
        except sr.RequestError:
            text = ""
        words = text.split()
        word_count = len(words)
        return {"text": text, "word_count": word_count}
    def process(self, duration=5):
        audio = self.record_audio(duration)
        result = self.analyze_audio(audio)
        return result

class VideoConference:
    def __init__(self, host="localhost", port=9999):
        self.host = host
        self.port = port
        self.server_socket = None
        self.client_socket = None
        self.is_server = False
        self.running = False
    def start_server(self):
        self.is_server = True
        self.server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.server_socket.bind((self.host, self.port))
        self.server_socket.listen(5)
        self.running = True
        self.client_socket, addr = self.server_socket.accept()
    def start_client(self):
        self.is_server = False
        self.client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.client_socket.connect((self.host, self.port))
        self.running = True
    def send_data(self, data):
        if self.client_socket:
            self.client_socket.sendall(data)
    def receive_data(self, buffer_size=4096):
        if self.client_socket:
            data = self.client_socket.recv(buffer_size)
            return data
        return None
    def stop(self):
        self.running = False
        if self.client_socket:
            self.client_socket.close()
        if self.server_socket:
            self.server_socket.close()
    def server_loop(self):
        while self.running:
            data = self.receive_data()
            if data:
                self.send_data(data)
            time.sleep(0.1)
```

```
    def client_loop(self, data_source):
        while self.running:
            data = data_source()
            if data:
                self.send_data(data)
            time.sleep(0.1)

class ReportingVisualization:
    def __init__(self):
        self.data = {}
    def add_data_point(self, key, value):
        if key in self.data:
            self.data[key].append(value)
        else:
            self.data[key] = [value]
    def generate_bar_chart(self):
        keys = list(self.data.keys())
        values = [sum(self.data[k]) for k in keys]
        plt.figure()
        plt.bar(keys, values)
        plt.xlabel("Categories")
        plt.ylabel("Values")
        plt.title("Bar Chart Report")
        plt.savefig("bar_chart_report.png")
        plt.close()
    def generate_line_chart(self):
        plt.figure()
        for key in self.data:
            plt.plot(self.data[key], label=str(key))
        plt.xlabel("Time")
        plt.ylabel("Values")
        plt.title("Line Chart Report")
        plt.legend()
        plt.savefig("line_chart_report.png")
        plt.close()
    def generate_pie_chart(self):
        keys = list(self.data.keys())
        values = [sum(self.data[k]) for k in keys]
        plt.figure()
        plt.pie(values, labels=keys, autopct="%1.1f%%")
        plt.title("Pie Chart Report")
        plt.savefig("pie_chart_report.png")
        plt.close()
    def export_data_to_csv(self, filename="report_data.csv"):
        with open(filename, "w", newline="") as csvfile:
            writer = csv.writer(csvfile)
            writer.writerow(["Category", "Values"])
            for key, values in self.data.items():
                writer.writerow([key, ",".join(map(str, values))])

class GestureRecognition:
    def __init__(self):
```

```
        self.gesture_labels = ["swipe_left", "swipe_right", "thumbs_up",
"thumbs_down", "open_hand", "closed_fist"]
        self.background_subtractor = cv2.createBackgroundSubtractorMOG2()
    def preprocess(self, frame):
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        blur = cv2.GaussianBlur(gray, (5, 5), 0)
        fg_mask = self.background_subtractor.apply(blur)
        return fg_mask
    def detect_gesture(self, frame):
        fg_mask = self.preprocess(frame)
        contours, _ = cv2.findContours(fg_mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        gesture = None
        if contours:
            largest_contour = max(contours, key=cv2.contourArea)
            if cv2.contourArea(largest_contour) > 1000:
                gesture = random.choice(self.gesture_labels)
        return gesture
    def analyze_frame(self, frame):
        gesture = self.detect_gesture(frame)
        if gesture:
            return {"gesture": gesture}
        else:
            return {"gesture": "none"}

def main():
    emotion_model_path = "emotion_model.h5"
    emotion_recognition = EmotionRecognition(emotion_model_path)
    voice_analyzer = VoiceAnalyzer()
    video_conf = VideoConference()
    reporting = ReportingVisualization()
    gesture_recognition = GestureRecognition()
    cap = cv2.VideoCapture(0)
    voice_results = []
    def voice_thread_func():
        while running_flag[0]:
            result = voice_analyzer.process(3)
            voice_results.append(result)
            time.sleep(2)
    running_flag = [True]
    voice_thread = threading.Thread(target=voice_thread_func)
    voice_thread.start()
    start_time = time.time()
    while time.time() - start_time < 60:
        ret, frame = cap.read()
        if not ret:
            continue
        emotion_results = emotion_recognition.analyze_frame(frame)
        gesture_result = gesture_recognition.analyze_frame(frame)
        reporting.add_data_point("emotion_count", len(emotion_results))
        if gesture_result["gesture"] != "none":
            reporting.add_data_point("gesture_count", 1)
        else:
```

```
                reporting.add_data_point("gesture_count", 0)
            cv2.imshow("Combined Surveillance", frame)
            if cv2.waitKey(1) & 0xFF == ord("q"):
                break
    running_flag[0] = False
    voice_thread.join()
    cap.release()
    cv2.destroyAllWindows()
    try:
        video_conf.start_server()
        server_thread = threading.Thread(target=video_conf.server_loop)
        server_thread.start()
        time.sleep(5)
        video_conf.stop()
        server_thread.join()
    except Exception:
        pass
    reporting.generate_bar_chart()
    reporting.generate_line_chart()
    reporting.generate_pie_chart()
    reporting.export_data_to_csv("final_report.csv")

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

Файл load_reference_texts.py

```
import os
import time
import math
import random

import json
import cv2
import numpy as np
from datetime import datetime

class PlagiarismChecker:

    def __init__(self, reference_texts_dir):
        self.reference_texts_dir = reference_texts_dir
        self.reference_texts = {}
        self.load_reference_texts()

    def load_reference_texts(self):
        for filename in os.listdir(self.reference_texts_dir):
            if filename.endswith(".txt"):

                filepath = os.path.join(self.reference_texts_dir, filename)
                with open(filepath, "r", encoding="utf-8") as f:
                    text = f.read()
                self.reference_texts[filename] = text

    def jaccard_similarity(self, text1, text2):
        set1 = set(text1.lower().split())
        set2 = set(text2.lower().split())
        intersection = set1.intersection(set2)
        union = set1.union(set2)
        if len(union) == 0:
            return 0.0
        return len(intersection) / len(union)

    def check_plagiarism(self, student_text):
        results = {}
        for ref_name, ref_text in self.reference_texts.items():
            similarity = self.jaccard_similarity(student_text, ref_text)
            results[ref_name] = similarity
        return results

    def generate_report(self, student_text, threshold=0.3):
        report = {}
        similarities = self.check_plagiarism(student_text)
        for ref_name, similarity in similarities.items():
            if similarity >= threshold:
                report[ref_name] = similarity
        return report

class BiometricIdentification:
```

```
    def __init__(self):
        self.enrolled_students = {}
    def extract_features(self, face_image):
        gray = cv2.cvtColor(face_image, cv2.COLOR_BGR2GRAY)
        resized = cv2.resize(gray, (32, 32))
        normalized = resized / 255.0
        features = normalized.flatten().tolist()
        return features
    def enroll_student(self, student_id, face_image):
        features = self.extract_features(face_image)

        self.enrolled_students[student_id] = features
    def cosine_similarity(self, vec1, vec2):
        dot = sum(a * b for a, b in zip(vec1, vec2))
        norm1 = math.sqrt(sum(a * a for a in vec1))

        norm2 = math.sqrt(sum(b * b for b in vec2))
        if norm1 == 0 or norm2 == 0:
            return 0.0
        return dot / (norm1 * norm2)
    def verify_student(self, student_id, face_image, threshold=0.8):
        if student_id not in self.enrolled_students:
            return False, 0.0
        features = self.extract_features(face_image)

       enrolled_features = self.enrolled_students[student_id]
        similarity = self.cosine_similarity(features, enrolled_features)
        return similarity >= threshold, similarity

class CheatingPrevention:
    def __init__(self):
        self.event_log = []
    def record_event(self, event_type, description, timestamp=None):
        if timestamp is None:
            timestamp = time.time()
        self.event_log.append({"event_type": event_type, "description":
description, "timestamp": timestamp})
    def analyze_events(self):

        risk_score = 0
        for event in self.event_log:
            if event["event_type"] == "window_switch":
                risk_score += 10

            elif event["event_type"] == "multiple_faces":
                risk_score += 15
            elif event["event_type"] == "suspicious_movement":
                risk_score += 5
            elif event["event_type"] == "keyboard_activity":

                risk_score += 3
        return risk_score
    def get_cheating_alert(self, threshold=20):
```

```
        score = self.analyze_events()
        if score >= threshold:
            return True, score
        return False, score
    def get_event_log(self):
        return self.event_log

class UserActivityAudit:
    def __init__(self):
        self.audit_log = []
    def log_action(self, action_type, details, timestamp=None):


        if timestamp is None:
            timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        self.audit_log.append({"action_type": action_type, "details": details,
"timestamp": timestamp})

    def get_audit_log(self):
        return self.audit_log

    def generate_audit_report(self):
        report_lines = []
        for entry in self.audit_log:
            line = f"{entry['timestamp']} - {entry['action_type']} -
{entry['details']}"
            report_lines.append(line)
        report = "\n".join(report_lines)
        return report

    def save_audit_report(self, filename="audit_report.txt"):
        report = self.generate_audit_report()
        with open(filename, "w", encoding="utf-8") as f:
            f.write(report)

class BehaviorAnalysis:

    def __init__(self):
        self.behavior_data = []

    def record_behavior(self, data_point):
        self.behavior_data.append(data_point)
    def analyze_behavior(self):
        if not self.behavior_data:
            return 0.0
        avg = sum(self.behavior_data) / len(self.behavior_data)
        return avg

    def detect_anomaly(self, threshold=0.5):
        avg = self.analyze_behavior()
        anomalies = []
        for idx, data in enumerate(self.behavior_data):
            if abs(data - avg) > threshold:
```

```
                anomalies.append((idx, data))
        return anomalies
    def generate_behavior_report(self):
        avg = self.analyze_behavior()

        anomalies = self.detect_anomaly()
        report = {"average": avg, "total_data_points": len(self.behavior_data),
"anomalies": anomalies}
        return report

def main():
    ref_dir = "reference_texts"


    os.makedirs(ref_dir, exist_ok=True)
    sample_text = "This is a sample reference text for plagiarism checking."
    with open(os.path.join(ref_dir, "sample1.txt"), "w", encoding="utf-8") as f:
        f.write(sample_text)
    plagiarism_checker = PlagiarismChecker(ref_dir)
    student_answer = "This is a sample student answer for checking plagiarism
against reference text."
    plagiarism_report = plagiarism_checker.generate_report(student_answer,

threshold=0.2)
    biometric = BiometricIdentification()
    dummy_face = np.random.randint(0, 255, (64, 64, 3), dtype=np.uint8)

    biometric.enroll_student("student_001", dummy_face)
    verify_result, similarity = biometric.verify_student("student_001",
dummy_face)
    cheating = CheatingPrevention()
    cheating.record_event("window_switch", "Student switched window at
unexpected time", time.time())
    cheating.record_event("multiple_faces", "Detected more than one face in
frame", time.time())

    cheating.record_event("keyboard_activity", "Unusual keyboard activity
detected", time.time())
    cheating_alert, cheating_score = cheating.get_cheating_alert(threshold=20)
    audit = UserActivityAudit()


    audit.log_action("login", "Student logged into the exam system")
    audit.log_action("start_test", "Student started the test")
    audit.log_action("submit_answer", "Student submitted an answer to question
1")
    audit_report = audit.generate_audit_report()
    audit.save_audit_report("audit_report.txt")

    behavior = BehaviorAnalysis()
    for i in range(50):
        behavior.record_behavior(random.uniform(0, 1))
    behavior_report = behavior.generate_behavior_report()
```

```
    output = {
        "plagiarism_report": plagiarism_report,
        "biometric_verification": {"result": verify_result, "similarity":
similarity},
        "cheating_alert": {"alert": cheating_alert, "score": cheating_score},
        "audit_report": audit_report,
        "behavior_report": behavior_report
    }

    with open("system_report.json", "w", encoding="utf-8") as f:
        json.dump(output, f, indent=4)
    print("System testing completed. Reports generated.")

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