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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи виявлення аномального навантаження серверних систем на основі машинного навчання”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-25М
ОПП «Комп’ютерні науки»
спеціальності F3 «Комп’ютерні науки»
_____ В.Р. Меркулов
«____» _____ 2026 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Р.М. Минайленко
«____» _____ 2026 р.

Нормоконтроль проекту
_____ А.С. Коваленко
«____» _____ 2026 р.

Рецензент _____
_____

Підписано КЕП
Центральноукраїнський національний технічний університет
МЕРКУЛОВ ВЛАДИСЛАВ РУСЛАНОВИЧ
Сертифікат: 21E4B79083A82E7F04000000050B24A02C1222F08
Дійсний з 24.11.2026 15:39:24 по 25.11.2027 01:59:59

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *другий (магістерський)*

Галузь знань F*"*Інформаційні технології*"*

Спеціальність F3*"*Комп'ютерні науки*"*

Освітньо-професійна (освітньо-наукова) програма *"*Комп'ютерні науки*"*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
"____" _____2026 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Меркулова Владислава Руслановича
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи виявлення аномального навантаження серверних систем на основі машинного навчання*

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

затверджені наказом вищого навчального закладу від 17.08.2026 року № 42-13

3. Строк подання роботи до захисту *21.12.2026 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є дослідження та програмна реалізація системи виявлення аномального навантаження серверних систем на основі машинного навчання*

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

1. Призначення та область використання.	*7. Заходи з охорони праці та техніки*
2.Перегляд аналогічних існуючих систем.	безпеки.
3. Опис і обґрунтування проектних рішень.	*8. Висновки*
4. Етапи програмування системи.	
5. Впровадження системи в промислову	
експлуатацію.	
6. Наукова новизна	

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

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

6. Консультанти з роботи, із зазначенням розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Марченко К.М., к.т.н., доцент	11.11.2026	13.11.2026

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	09.10.2026 р.	
2.	Постановка задачі, оформлення ТЗ	12.10.2026 р.	
3.	Розробка моделі компонента	15.10.2026 р.	
4.	Розробка структур даних	19.10.2026 р.	
5.	Розробка алгоритмів зв’язку та відображення	21.10.2026 р.	
6.	Програмування алгоритмів	25.10.2026 р.	
7.	Розрахунки з охорони праці та техніки безпеки	11.10.2026 р.	
8.	Оформлення ПЗ	13.11.2026 р.	
9.	Попередній захист роботи	28.11.2026 р.	

Дата видачі завдання
«__»_____2026 р.

Підпис керівника
_____ Минайленко Р. М.
(прізвище та ініціали)
Завдання прийнято до виконання
«__»_____2026 р.

Підпис здобувача вищої освіти
_____ Меркулов В. Р.
(прізвище та ініціали)

АНОТАЦІЯ

Дослідження та програмна реалізація системи виявлення аномального навантаження серверних систем на основі машинного навчання. F3 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи виявлення аномального навантаження серверних систем на основі машинного навчання.

Метою розробки є дослідження та програмна реалізація системи виявлення аномального навантаження серверних систем на основі машинного навчання.

Об'єктом дослідження є процес виявлення аномального навантаження серверних систем на основі машинного навчання.

Предметом дослідження є методи виявлення аномального навантаження серверних систем на основі машинного навчання.

Методами дослідження є системний аналіз для вивчення принципів функціонування систем, порівняльний аналіз для зіставлення результатів, статистичний аналіз для опрацювання показників, методи машинного навчання.

Результат роботи – програмна реалізація системи виявлення аномального навантаження серверних систем на основі машинного навчання.

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'ютерні науки, аномальне навантаження, серверні системи, машинне навчання

ABSTRACT

Merkulov V.R. Research and software implementation of a system for detecting anomalous server systems load based on machine learning. F3 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

In this final qualification work for the second (master's) level of higher education, software has been developed, which is intended for a system for detecting anomalous server systems load based on machine learning.

The purpose of the development is the research and software implementation of a system for detecting anomalous server systems load based on machine learning.

The object of the research is the process of detecting anomalous server systems load based on machine learning.

The subject of the research is the methods of detecting anomalous server systems load based on machine learning.

The research methods are systems analysis for studying the principles of system functioning, comparative analysis for contrasting results, statistical analysis for processing metrics, machine learning methods.

The result of the work is a software implementation of a system for detecting anomalous server systems load based on machine learning.

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 the software are provided

A user-friendly interface has been developed. Instructions for working with the software are provided.

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

The program was developed in the Python environment.

Keywords: computer science, anomalous load, server systems, machine learning

ВІДОМІСТЬ ПРОЕКТУ

		Текстові документи			
1	А4	Пояснювальна записка	96	–	
2	А4	Робоча програма	17	–	Додаток А
		Графічні документи			
3	А4	Наукова новизна	1	–	
4	А4	Структурна схема	1	–	
5	А4	Функціональна схема роботи системи	1	–	
6	А4	Діаграма взаємодії процесів	1	–	
7	А4	Блок-схема основної програми	1	–	
8	А4	Блок-схема роботи підпрограми	1	–	

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 8
ВСТУП..... 9
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 11
1.1 Призначення системи..... 11
1.2 Область використання 13
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 14
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 14
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування 38
2.3 Розгорнута постановка завдання 44
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 47
3.1 Опис функціонування системи 47
3.2 Розробка структурної схеми..... 50
3.3 Розробка функціональної схеми 52
3.4 Розробка діаграми процесів 55
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ ТА ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ І ПРОГРАМНИХ РІШЕНЬ 57
4.1 Розробка блок-схем та опис алгоритмів функціонування системи 57
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 69
6 НАУКОВА НОВИЗНА 79
7 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ 80
7.1 Вступ..... 80
7.2 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста ... 81
7.3 Розробка заходів з поліпшення стану охорони праці 83
7.4 Техніка безпеки та протипожежна профілактика 84
7.5 Розрахункова частина 85
8 ОСНОВНІ ВИСНОВКИ 89
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 90

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення

IF – Isolation Forest

IT – інформаційні технології

AWS – Amazon Web Services

ITSI – Information Technology Service Intelligence

IBM – International Business Machines

SL1 – ScienceLogic 1

AIOps – Artificial Intelligence for IT Operations

API – Application Programming Interface

DB – Database

FMS – Flexible Monitoring System

SMS – Short Message Service

ОС – операційна система

JSON – JavaScript Object Notation

CSV – Comma-Separated Values

SMTP – Simple Mail Transfer Protocol

JWT – JSON Web Token

TLS – Transport Layer Security

HTML – HyperText Markup Language

VSC – Visual Studio Code

Gmail – Google Mail

ДБН – державні будівельні норми

ДСН – державні санітарні норми

ВСТУП

Розвиток цифрових технологій супроводжується збільшенням обсягів інформації, розширенням інфраструктури та зростанням навантаження на серверні системи. Функціонування підприємств залежить від надійності їхнього серверного середовища та ресурсів. Будь-які порушення цілісності інформації та стабільності роботи призводять до збитків, затримок у роботі та втрати важливих даних [1, 4].

Впровадження систем моніторингу серверів є важливим завданням. Системи моніторингу часто використовують пороговий підхід, за якого перевантаження визначається порівнянням поточних показників із попередньо встановленими значеннями. Такий підхід є простим у реалізації, однак не завжди дозволяє виявляти нетипові поєднання показників, які не перевищують заданих порогів. Використання машинного навчання дає змогу аналізувати різні взаємозв’язки між показниками та визначати відповідні відхилення від поведінки системи.

Одним із напрямів вирішення цієї проблеми є застосування моделей машинного навчання без учителя. Алгоритм Isolation Forest демонструє високу ефективність під час ізоляції нетипових спостережень. Застосування цього алгоритму до показників використання серверних ресурсів дозволяє дослідити можливості виявлення аномального навантаження та порівняти результати з пороговим аналізом [1, 2].

Актуальність теми полягає в необхідності підвищення ефективності виявлення аномального навантаження серверних систем шляхом використання методів машинного навчання, що дозволяють аналізувати використання ресурсів та виявляти нетипові стани системи.

Метою розробки є дослідження та програмна реалізація системи виявлення аномального навантаження серверних систем на основі машинного навчання.

Для досягнення мети були визначені наступні **завдання**:

– провести аналіз існуючих систем моніторингу та навантаження;

– дослідити функціонування алгоритму Isolation Forest;

– реалізувати модуль моніторингу, аналізу та виявлення аномалій;

– впровадити модуль сповіщення, логування та конфігурацію;

– виконати порівняння результатів та провести тестування системи.

Об'єктом дослідження є процес виявлення аномального навантаження серверних систем на основі машинного навчання.

Предметом дослідження є методи виявлення аномального навантаження серверних систем на основі машинного навчання.

Методами дослідження є системний аналіз для вивчення принципів функціонування систем, порівняльний аналіз для зіставлення результатів, статистичний аналіз для опрацювання показників, методи машинного навчання.

Наукова новизна одержаних результатів полягає у тому, що:

1) Удосконалено підхід до виявлення аномального навантаження серверних систем шляхом використання алгоритму Isolation Forest для аналізу показників використання центрального процесора, оперативної пам'яті та дискового простору, що дозволяє виявляти нетипові зміни у сукупності метрик та визначати приховані аномалії.

2) Удосконалено процес обробки даних моніторингу та процес інформування про аномальне навантаження серверної системи на основі використання результатів аналізу показників для формування повідомлень про виявлені перевантаження та аномалії.

Практичне значення отриманих результатів полягає у створенні системи, яка забезпечує збір показників використання серверних ресурсів, їх збереження, аналіз, виявлення аномальних станів та формування результатів для подальшого оцінювання навантаження.

Апробація результатів роботи: питання застосування штучного інтелекту, ефективного використання обчислювальних ресурсів та моніторингу розглядалися в наукових публікаціях і дослідницьких роботах [3, 4, 5].

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система, що розробляється в межах даної роботи, є програмним забезпеченням системи виявлення аномального навантаження серверних систем на основі машинного навчання. Основне призначення системи полягає у зборі та аналізі показників використання серверних ресурсів, виявленні перевантажень і нетипових станів, а також забезпеченні можливості подальшого реагування на виявлені відхилення.

Виявлення аномального навантаження здійснюється шляхом аналізу показників використання центрального процесора, оперативної пам’яті та дискового простору. Для цього застосовуються пороговий метод аналізу та алгоритм машинного навчання Isolation Forest, який дозволяє визначати нетипові спостереження на основі зібраних даних.

Система виконує такі основні функції:

1) Моніторинг – збір даних про стан сервера, таких як навантаження на процесор, використання оперативної пам’яті, заповнення дискового простору, обсяг мережевої активності.

2) Аналіз даних – обробка отриманих метрик із метою виявлення навантажень або критичних станів системи. На основі порогових значень система визначає, чи перевищують поточні показники допустимі межі.

3) Виявлення аномалій – застосування алгоритму машинного навчання Isolation Forest для аналізу показників використання серверних ресурсів та визначення нетипових станів системи.

4) Розподіл ресурсів – виконання дій щодо оптимізації використання серверних ресурсів та зменшення навантаження на систему.

5) Система сповіщень – надсилання повідомлень у разі виявлення перевантаження або критичного стану серверної системи.

6) Логування – запис показників використання ресурсів, результатів аналізу та основних подій у лог-файли, що дозволяє відстежувати історію роботи системи.

7) Безпека – використання механізмів автентифікації та авторизації для обмеження доступу до окремих функцій системи.

8) Управління конфігурацією – можливість налаштування параметрів моніторингу, таких як інтервал збору даних і порогові значення для забезпечення функціонування системи.

9) Тестування – перевірка коректності роботи програмних модулів та основних функцій системи.

10) Документація – відображення інструкції для зручного використання системи.

Особливістю системи є використання результатів машинного навчання спільно з результатами порогового аналізу. Пороговий метод визначає перевищення заздалегідь установлених значень використання ресурсів, тоді як Isolation Forest аналізує сукупність показників і виявляє нетипові спостереження. Порівняння отриманих результатів дає змогу визначати випадки, у яких обидва методи виявляють відхилення, а також випадки розбіжностей між ними.

Основні переваги системи:

- поєднання двох методів аналізу;
- виявлення відхилень від нормальної роботи системи;
- повторне використання навченої моделі;
- фіксування результатів у лог-файлах;
- динамічне налаштування параметрів.

Таким чином, розроблений модуль не обмежується перевіркою окремих показників за порогами, а використовує модель машинного навчання для виявлення нетипових станів навантаження. Його застосування у складі системи забезпечує додатковий спосіб оцінювання стану серверних ресурсів. Поєднання порогового аналізу з алгоритмом Isolation Forest дозволяє досліджувати відмінності між визначенням перевантаження та виявленням аномалій.

1.2 Область використання

Розроблена система може використовуватися в різних сферах діяльності, де існує необхідність контролю навантаження на обладнання з метою своєчасного виявлення перевантажень, аналізу використання ресурсів та визначення відхилень у його роботі.

Основні сфери застосування [6]:

1) Організації з великою IT-інфраструктурою – банки, оператори зв'язку, страхові компанії, які використовують серверне обладнання для обробки великих обсягів інформації. Моніторинг навантаження дозволяє контролювати використання ресурсів та виявляти нетипові стани серверних систем.

2) Центри обробки даних – забезпечують функціонування великої кількості серверів, для яких важливими є стабільність та ефективне використання ресурсів. Застосування методів машинного навчання може доповнювати моніторинг шляхом виявлення аномальних показників навантаження.

3) Державні структури – використовують інформаційні системи, бази даних та електронні сервіси, для стабільної роботи яких необхідний контроль стану серверного обладнання.

4) Малі та середні підприємства – можуть використовувати систему для автоматизації збору показників навантаження, аналізу стану серверів та отримання сповіщень про перевищення встановлених порогових значень.

5) Хмарні компанії – використовують серверні ресурси для надання обчислювальних послуг. Застосування системи моніторингу та виявлення аномалій може сприяти визначенню навантаження на окремі сервери.

Розроблена система може застосовуватися в різних сферах діяльності, де необхідно здійснювати моніторинг навантаження та виявляти аномальні стани. Її використання дозволяє виконувати збір, аналіз показників серверних ресурсів, порівнювати результати та формувати інформацію для подальшого оцінювання стану серверної системи. Завдяки цьому забезпечується стабільність та ефективність роботи серверів [6].

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем

Аналіз існуючих систем показує, що існує велика кількість різних інструментів для моніторингу, аналізу навантаження та виявлення аномальних станів із використанням методів машинного навчання [7].

Netdata – система моніторингу серверних ресурсів у реальному часі, яка забезпечує збір та візуалізацію показників використання процесора, оперативної пам’яті та дискового простору. Система використовує алгоритм машинного навчання k-means для автоматичного виявлення аномалій. Вигляд інтерфейсу Netdata має вид [8]:



Рисунок 2.1 – Вид інтерфейсу Netdata

Переваги [9]:

- простота використання;
- налаштування;
- працює у реальному часі.

Недоліки [9]:

– недостатність документації;

– ресурсомісткість;

– обмежена інтеграція.

OpenSearch Anomaly Detection – інструмент для виявлення аномалій у часових рядах за допомогою машинного навчання. Система використовує алгоритм Random Cut Forest для аналізу показників, визначення відхилень та оцінювання аномальності отриманих даних. Результати аналізу можна переглядати через OpenSearch Dashboards. Вигляд інтерфейсу OpenSearch має вид [10]:

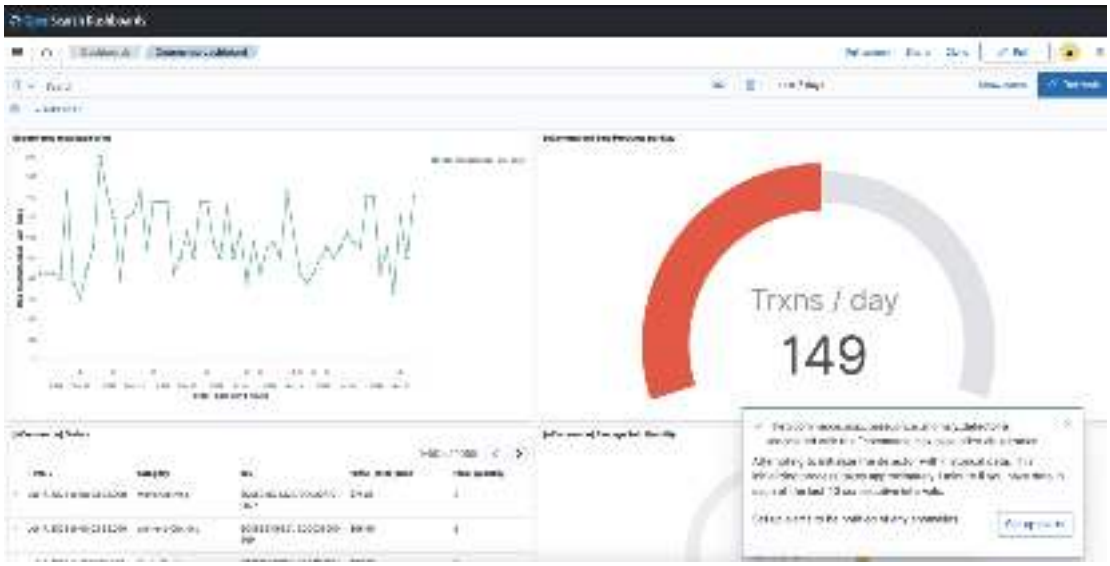

Рисунок 2.2 – Вид інтерфейсу OpenSearch

Переваги [11]:

– підтримка AWS;

– рівень безпеки;

– відкритий код.

Недоліки [12]:

– недостатність документації;

– ресурсомісткість;

– висока вартість.

Dynatrace – платформа моніторингу програмного забезпечення та серверів, яка забезпечує аналіз продуктивності й виявлення відхилень у роботі систем. Для визначення аномалій використовуються сформовані показники та порогові значення. Платформа дозволяє відстежувати стан серверів, застосунків і різних сервісів. Вигляд інтерфейсу Dynatrace має вид [13]:

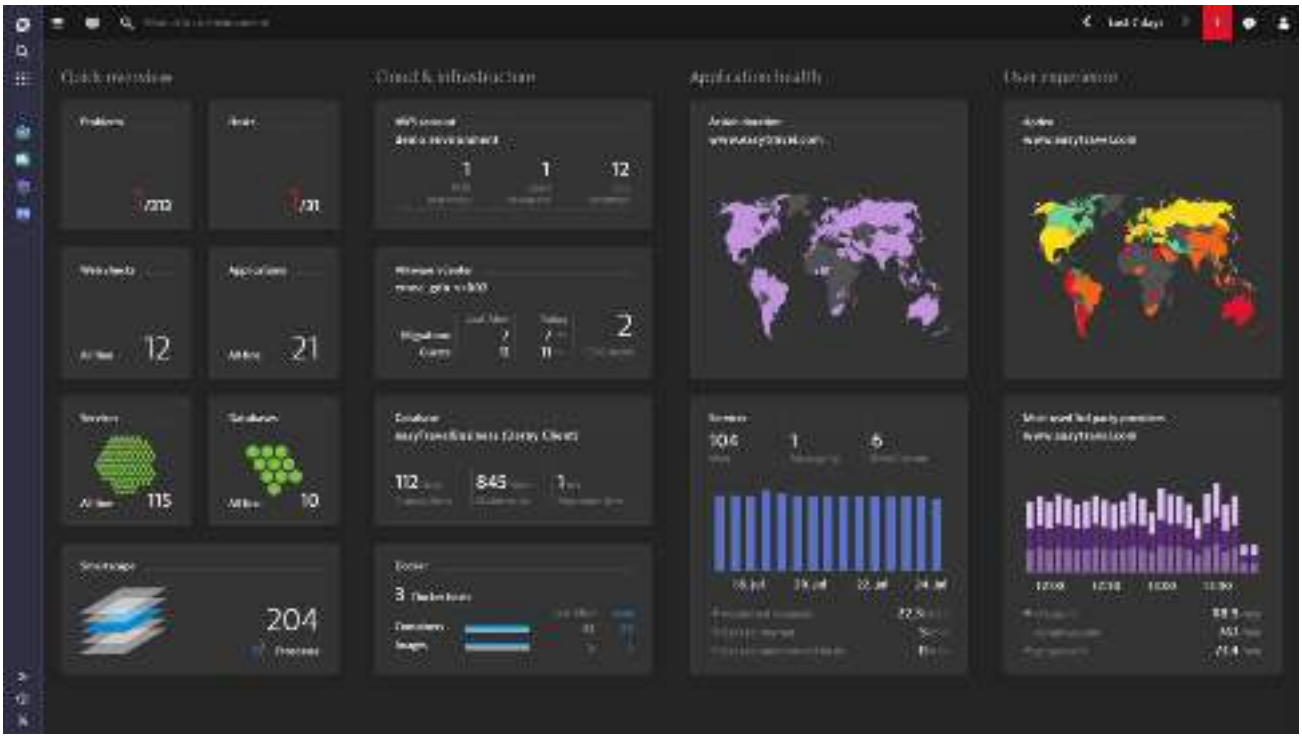

Рисунок 2.3 – Вид інтерфейсу Dynatrace

Переваги [14]:

– простий інтерфейс;
– гнучкість;
– ефективність;
– документація;
– масштабованість.

Недоліки [15]:

– вартість навчання;
– налаштування;
– конфігурація.

Datadog Watchdog – інструмент платформи Datadog, призначений для автоматичного виявлення збоїв серверів та програмних застосунків. Система аналізує зібрані показники, визначає відхилення та надає інформацію для дослідження можливих причин виникнення проблем. Вигляд інтерфейсу Datadog Watchdog має вид [16]:

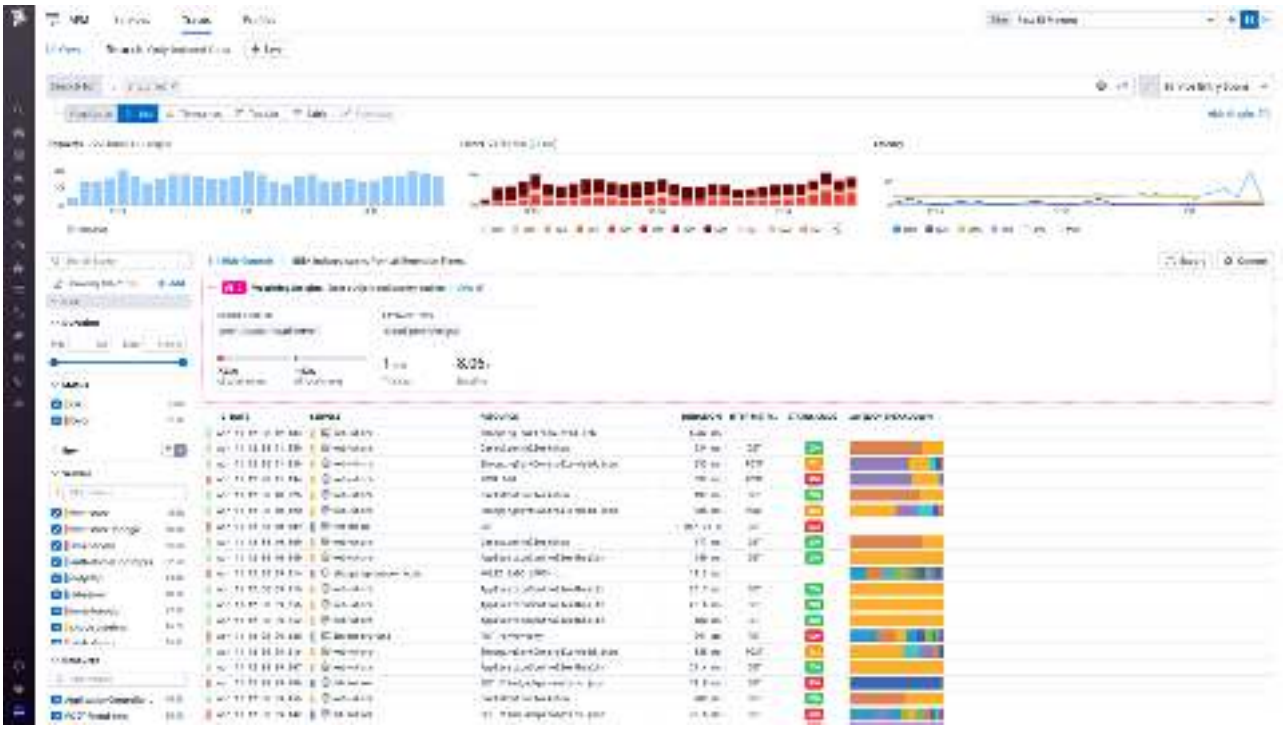

Рисунок 2.4 – Вид інтерфейсу Datadog Watchdog

Переваги [17]:

– простота використання;
– налаштування;
– працює у реальному часі;
– зручний інтерфейс.

Недоліки [18]:

– висока вартість;
– зберігання логів;
– ресурсомісткість;
– недостатність документації.

New Relic – платформа моніторингу продуктивності серверів, програмних застосунків та інформаційних систем. Вона підтримує виявлення аномалій шляхом аналізу показників та визначення відхилень. Користувач може налаштовувати чутливість виявлення аномалій та отримувати сповіщення про зміни. Вигляд інтерфейсу New Relic має вид [19]:



Рисунок 2.5 – Вид інтерфейсу New Relic

Переваги [20]:

– простий інтерфейс;

– працює у реальному часі;

– аналітика даних.

Недоліки [21]:

– налаштування;

– технічна підтримка;

– висока вартість.

Grafana Cloud Machine Learning – набір інструментів для аналізу часових рядів та використання машинного навчання. Він дозволяє досліджувати поведінку показників, прогнозувати їх зміни та визначати відхилення. Результати аналізу можуть відображатися на інформаційних панелях Grafana у вигляді графіків. Вигляд інтерфейсу Grafana Cloud має вид [22]:



Рисунок 2.6 – Вид інтерфейсу Grafana Cloud

Переваги [23]:

– гнучкість налаштування;
– підтримка плагінів;
– працює у реальному часі;
– рівень безпеки;
– оновлення.

Недоліки [23]:

– розуміння інтерфейсу;
– технічна підтримка;
– сповіщення;
– ресурсомісткість;
– недостатність документації.

Elastic Observability – платформа для моніторингу серверів, застосунків та аналізу лог-файлів. Вона підтримує використання машинного навчання для виявлення аномалій у часових рядах. Результати моніторингу й аналізу відображаються через інтерфейс Kibana. Вигляд інтерфейсу Elastic Observability має вид [24]:



Рисунок 2.7 – Вид інтерфейсу Elastic Observability

Переваги [25]:

– моніторинг у реальному часі;

– легкість використання.

Недоліки [25]:

– зберігання логів;

– висока вартість.

Splunk IT Service Intelligence – платформа для моніторингу стану інформаційних сервісів та аналізу продуктивності. Вона підтримує використання машинного навчання для визначення аномалій. Вигляд інтерфейсу Splunk ITSI має вид [26]:



Рисунок 2.8 – Вид інтерфейсу Splunk IT Service Intelligence

Переваги [27]:

– аналітика даних;

– сповіщення.

Недоліки [27]:

– вартість ліцензування.

Amazon CloudWatch Anomaly Detection – інструмент хмарної платформи Amazon Web Services, призначений для виявлення відхилень у показниках моніторингу. Вигляд інтерфейсу Amazon CloudWatch Anomaly Detection має вид [28]:



Рисунок 2.9 – Вид інтерфейсу Amazon CloudWatch Anomaly Detection

Переваги [29]:

– гнучкість налаштування;

– сповіщення;

– управління логами.

Недоліки [29]:

– розуміння інтерфейсу.

– масштабованість.

Sumo Logic – хмарна платформа для збору, обробки та аналізу лог-файлів. Вона підтримує інструменти машинного навчання для аналізу даних, виявлення закономірностей та дослідження проблем. Вигляд інтерфейсу Sumo Logic має вид [30]:



Рисунок 2.10 – Вид інтерфейсу Sumo Logic

Переваги [31]:

– легкість використання;

– конфігурація;

– моніторинг у реальному часі.

Недоліки [31]:

– висока вартість;

– продуктивність.

IBM Instana – платформа моніторингу програмних застосунків та сервісів. Система забезпечує збір показників, аналіз взаємозв'язків між компонентами та виявлення проблем у роботі застосунків. Вигляд інтерфейсу IBM Instana має вид [32]:



Рисунок 2.11 – Вид інтерфейсу IBM Instana

Переваги [33]:

– продуктивність;

– працює у реальному часі;

– налаштування.

Недоліки [33]:

– розуміння інтерфейсу;

– технічна підтримка.

Cisco Observability Platform – платформа, яка збирає дані з усіх програм, мереж та серверів. Платформа дозволяє аналізувати показники продуктивності та використовувати результати для оцінювання стану системи. Вигляд інтерфейсу Cisco Observability Platform має вид [34]:



Рисунок 2.12 – Вид інтерфейсу Cisco Observability Platform

Переваги [35]:

– аналітика даних;

– рівень безпеки;

– м моніторинг у реальному часі.

Недоліки [36]:

– налаштування;

– продуктивність.

SolarWinds Observability – платформа моніторингу серверів, мережевого обладнання, застосунків та хмарних середовищ. Вона забезпечує збір показників використання ресурсів, візуалізацію продуктивності та формування сповіщень. Вигляд інтерфейсу SolarWinds Observability має вид [37]:

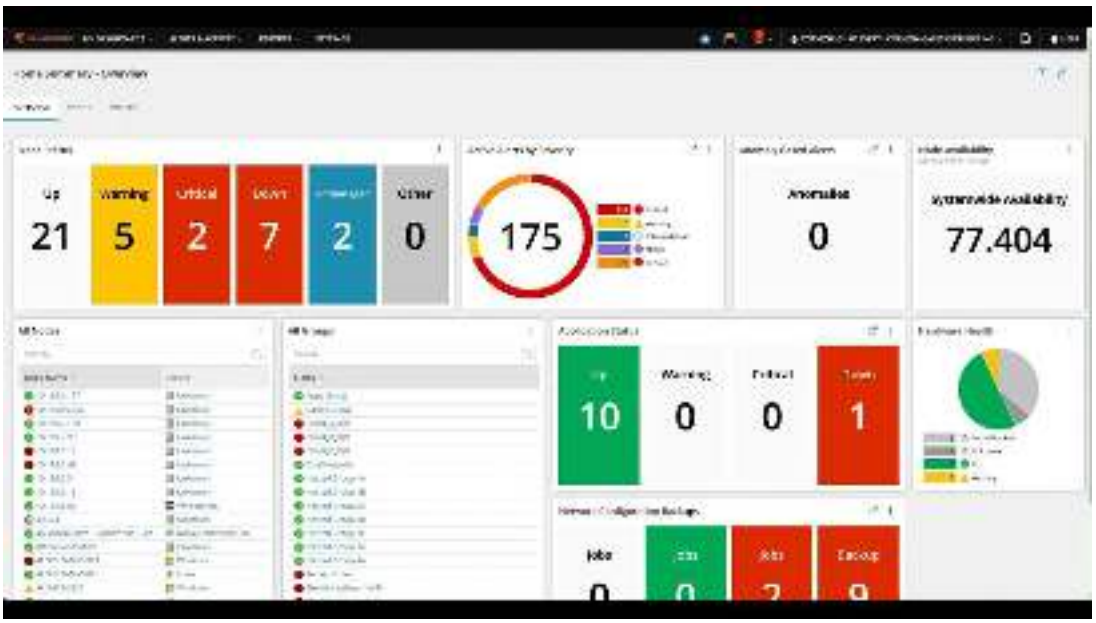

Рисунок 2.13 – Вид інтерфейсу SolarWinds Observability

Переваги [38]:

– інтуїтивно зрозумілий інтерфейс;

– інтеграція з сервісами.

Недоліки [38]:

– висока вартість;

– налаштування.

LogicMonitor – платформа моніторингу програм, яка забезпечує збір показників роботи серверів, мережевого обладнання та хмарних ресурсів. Вигляд інтерфейсу LogicMonitor має вид [39]:



Рисунок 2.14 – Вид інтерфейсу LogicMonitor

Переваги [40]:

– технічна підтримка;

– легкість використання;

– налаштування.

Недоліки [40]:

– обмеженість функцій;

– конфігурація;

– висока вартість.

ScienceLogic SL1 – платформа для моніторингу та управління станом системи. Вона забезпечує збір даних із серверів, мережевого обладнання та програмних сервісів. Вигляд інтерфейсу ScienceLogic SL1 має вид [41]:



Рисунок 2.15 – Вид інтерфейсу ScienceLogic SL1

Переваги [42]:

– універсальність;

– гнучкість.

Недоліки [42]:

– висока вартість;

– налаштування.

BigPanda – платформа AIOps для аналізу сповіщень. Вона об’єднує події з різних систем моніторингу та встановлює зв’язки між ними. Вигляд інтерфейсу BigPanda має вид [43]:

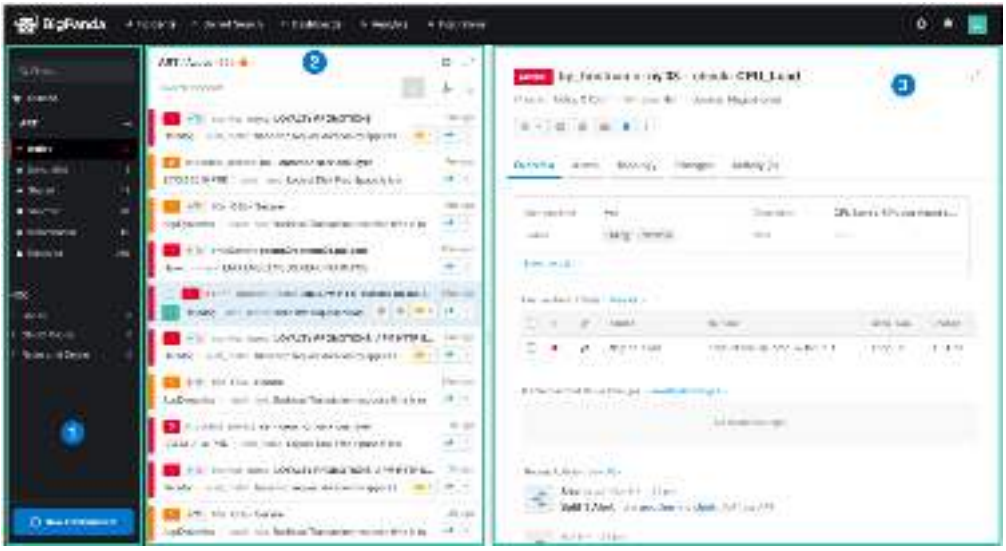

Рисунок 2.16 – Вид інтерфейсу BigPanda

Переваги [44]:

– звітність;

– інтеграція з сервісами;

– сповіщення.

Недоліки [44]:

– налаштування;

– недостатність документації.

Moogsoft – платформа AIOps, призначена для аналізу подій та управління інцидентами. Вигляд інтерфейсу Moogsoft має вид [45]:



Рисунок 2.17 – Вид інтерфейсу Moogsoft

Переваги [46]:

– інтуїтивно зрозумілий інтерфейс;

– сповіщення;

– інтеграція з сервісами.

Недоліки [46]:

– технічна підтримка;

– нестабільність роботи.

ManageEngine Applications Manager – система моніторингу серверів та програмних застосунків. Вона забезпечує контроль використання процесора, оперативної пам’яті, дискового простору та інших показників. Вигляд інтерфейсу ManageEngine Applications Manager має вид [47]:



Рисунок 2.18 – Вид інтерфейсу ManageEngine Applications Manager

Переваги [48]:

– моніторинг у реальному часі;

– легкість використання;

– гнучкість налаштування.

Недоліки [48]:

– обмеженість функцій;

– інтеграція з сервісами;

– розуміння інтерфейсу.

Site24x7 – платформа для моніторингу серверів, застосунків та мережевих ресурсів. Вона забезпечує збір показників та виявлення аномалій. Вигляд інтерфейсу Site24x7 має вид [49]:

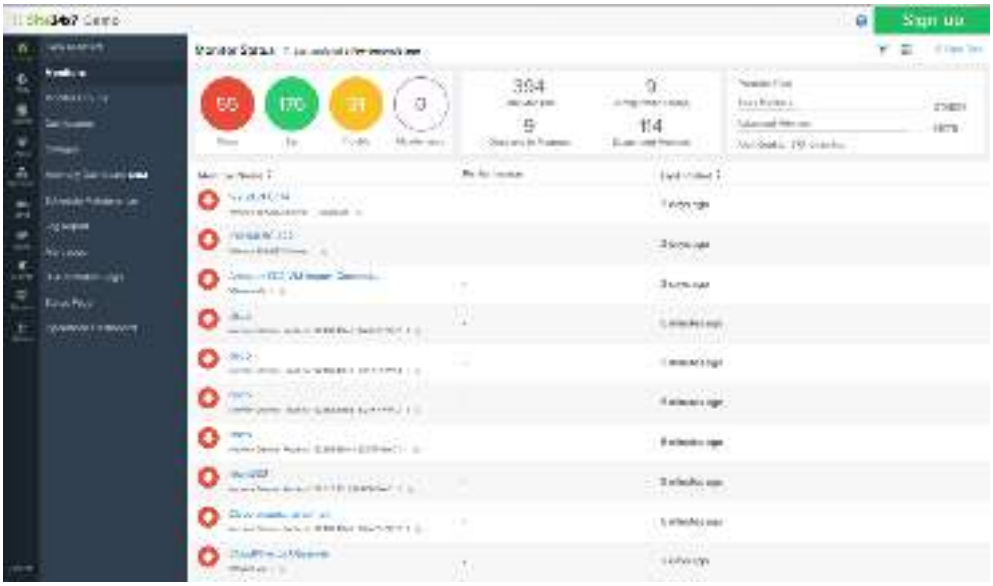

Рисунок 2.19 – Вид інтерфейсу Site24x7

Переваги [50]:

– моніторинг у реальному часі;

– легкість використання.

Недоліки [50]:

– сповіщення;

– висока вартість.

Sematext Cloud – платформа для моніторингу серверів, програмних застосунків та аналізу лог-файлів. Вигляд інтерфейсу Sematext Cloud має вид [51]:

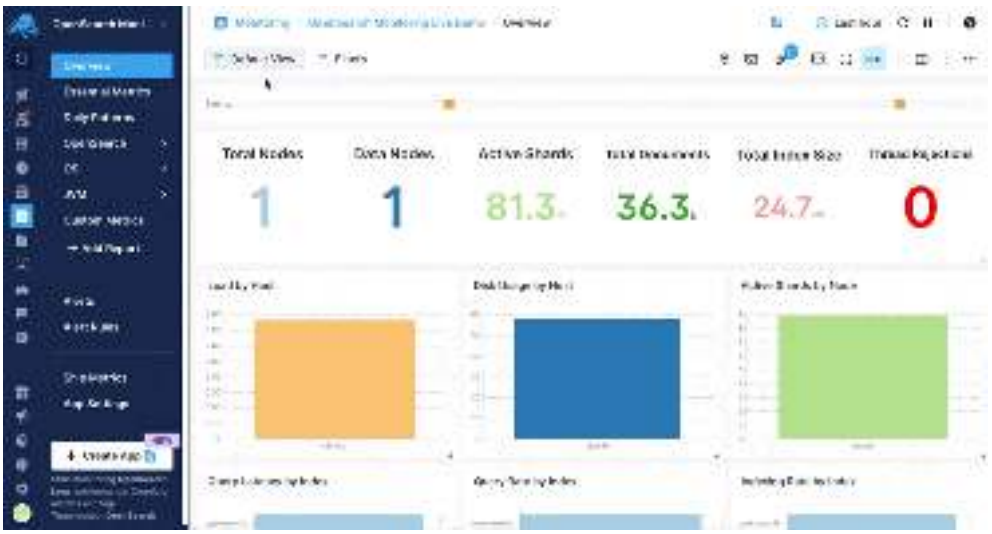

Рисунок 2.20 – Вид інтерфейсу Sematext Cloud

Переваги [52]:

– інтеграція з сервісами;

– відкритий код;

– підтримка API Elasticsearch та InfluxDB.

Недоліки [52]:

– висока вартість;

– конфігурація;

– обмеженість функцій.

Honeycomb – платформа для аналізу даних і дослідження продуктивності систем. Вона дозволяє аналізувати запити, затримки та помилки роботи застосунків. Вигляд інтерфейсу Honeycomb має вид [53]:

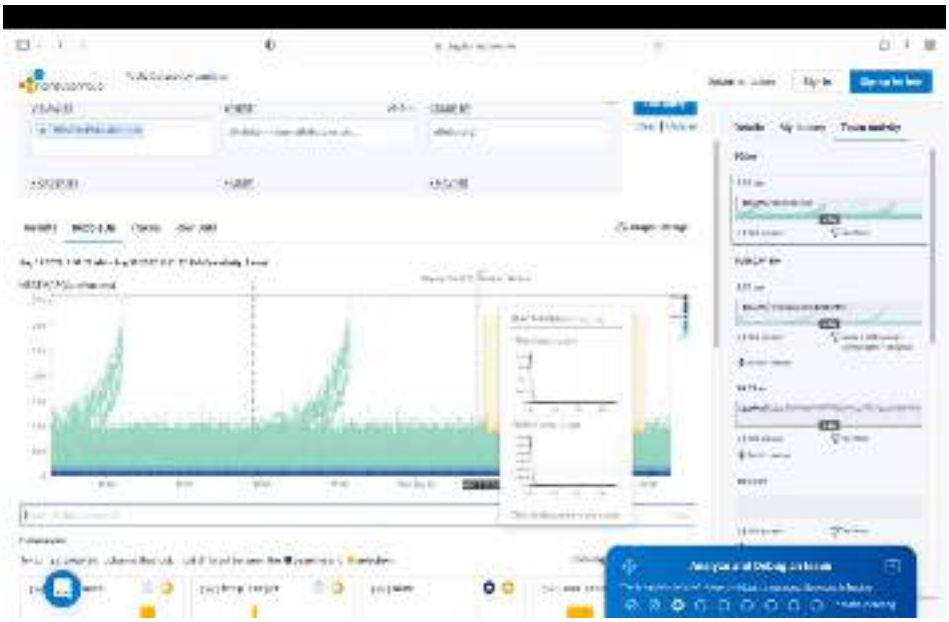

Рисунок 2.21 – Вид інтерфейсу Honeycomb

Переваги [54]:

– легкість використання;

– аналітика даних;

– моніторинг у реальному часі;

– технічна підтримка;

– гнучкість.

Недоліки [54]:

– висока вартість;

– налаштування;

– обмеженість функцій.

VictoriaMetrics – система зберігання та аналізу часових рядів, яка використовується для обробки показників моніторингу серверів. Для виявлення аномалій застосовується компонент VictoriaMetrics Anomaly Detection. Вигляд інтерфейсу VictoriaMetrics має вид [55]:



Рисунок 2.22 – Вид інтерфейсу VictoriaMetrics

Переваги [56]:

– ефективність;

– легкість використання;

– налаштування;

– гнучкість.

Недоліки [56]:

– візуалізація;

– обмеженість функцій;

– технічна підтримка.

EventSentry – інструмент моніторингу для Windows, який у реальному часі відстежує лог-файли та системні події. Додаток складається з консолі керування, служби Heartbeat та модуля веб-звітності. Вигляд інтерфейсу EventSentry має вид [57]:



Рисунок 2.23 – Вид інтерфейсу EventSentry

Переваги [58]:

– ефективність;

– сповіщення;

– технічна підтримка.

Недоліки [58]:

– зберігання логів;

– налаштування;

– візуалізація.

Pandora FMS – система моніторингу, яка призначена для відстеження стану продуктивності та безпеки серверів. Вона виявляє пристрої та визначає топологію мережі. Архітектура системи є відкритою та гнучкою. Вигляд інтерфейсу Pandora FMS має вид [59]:



Рисунок 2.24 – Вид інтерфейсу Pandora FMS

Переваги [60]:

– моніторинг у реальному часі;

– відкритий код;

– визначення топології мережі;

– підтримка протоколів;

– гнучкість налаштування;

– звітність.

Недоліки [60]:

– навчання системи;

– конфігурація;

– обмеженість функцій;

– відсутність API;

– технічна підтримка.

SigNoz – платформа моніторингу програмних застосунків та серверів з відкритим кодом. Вона забезпечує збір і аналіз метрик, дозволяє створювати інформаційні панелі й налаштовувати сповіщення. Система може використовуватися для виявлення відхилень. Вигляд інтерфейсу SigNoz має вид [61]:

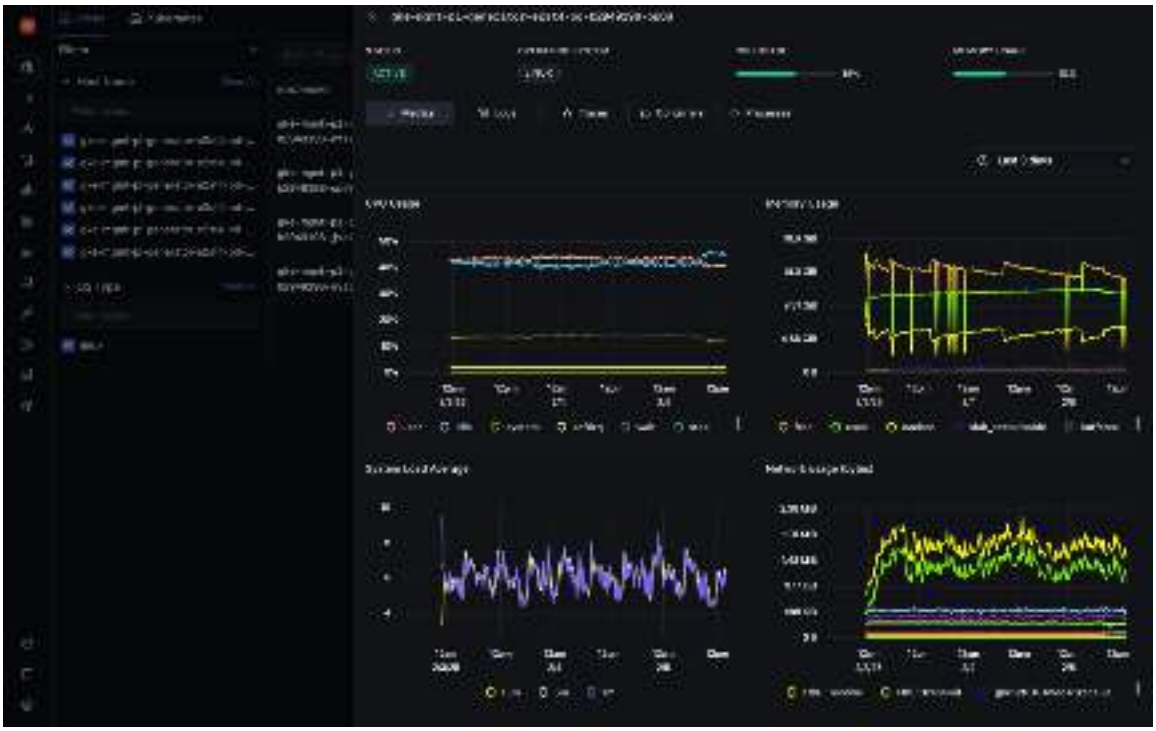

Рисунок 2.25 – Вид інтерфейсу SigNoz

Переваги [61]:

– зберігання логів;

– сповіщення;

– простота інтерфейсу;

– налаштування;

– відкритий код.

Недоліки [61]:

– обмеженість функцій;

– конфігурація;

– висока вартість;

– ресурсомісткість;

– оновлення.

Cabot – це платформа для моніторингу з відкритим вихідним кодом. Вона поєднує в собі функції таких сервісів, як PagerDuty, Server Density, Pingdom та Nagios, забезпечуючи надсилання сповіщень через SMS. Вигляд інтерфейсу Cabot має вид [62]:

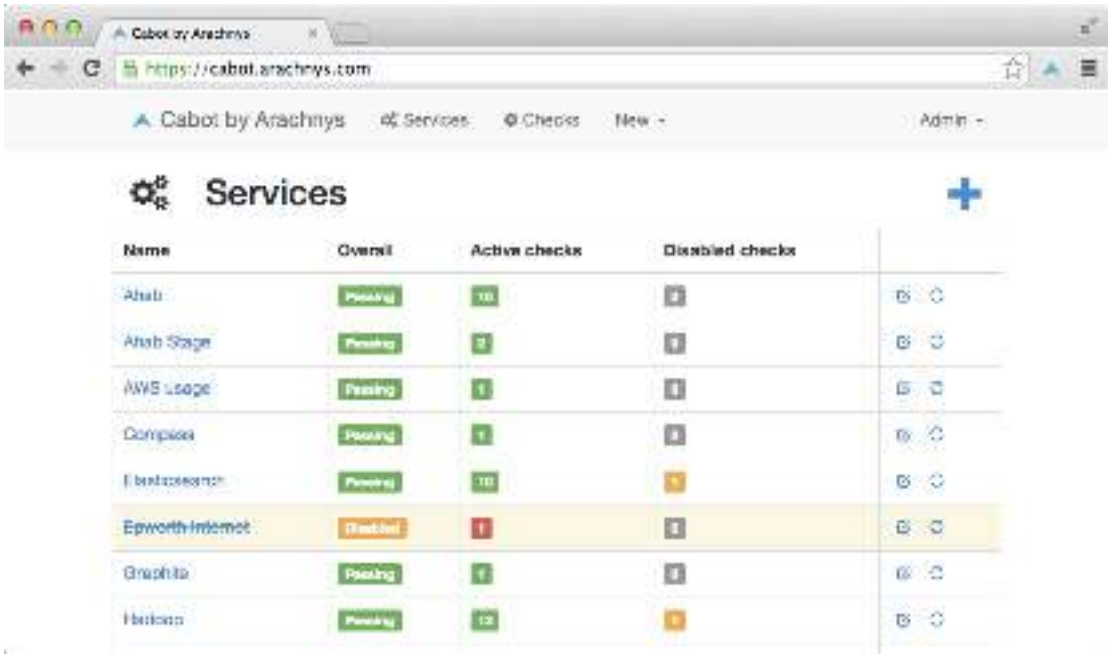

Рисунок 2.26 – Вид інтерфейсу Cabot

Програмне забезпечення Cabot призначене для конфігурації пристроїв у мережі EasyBUS, перевірки їхньої роботи, а також керування функціями. Завдяки інтерфейсу система дозволяє здійснювати налаштування в автономному режимі. Окрім цього, Cabot забезпечує нагляд за системою в реальному часі та підтримує швидку заміну модулів, що робить його зручним інструментом для моніторингу та налаштування [62].

Переваги [63]:

– легкість використання;
– відкритий код;
– гнучкість;
– простота інтерфейсу.

Недоліки [63]:

– обмеженість функцій;
– конфігурація;
– технічна підтримка.
– ресурсомісткість;
– недостатність документації.

Opsview – платформа для моніторингу хмарних технологій та різної IT-інфраструктури. Вона дозволяє копіювати сервіси для швидкого налаштування моніторингу, а також допомагає мінімізувати кількість непотрібних сповіщень. Вигляд інтерфейсу Opsview має вид [64]:

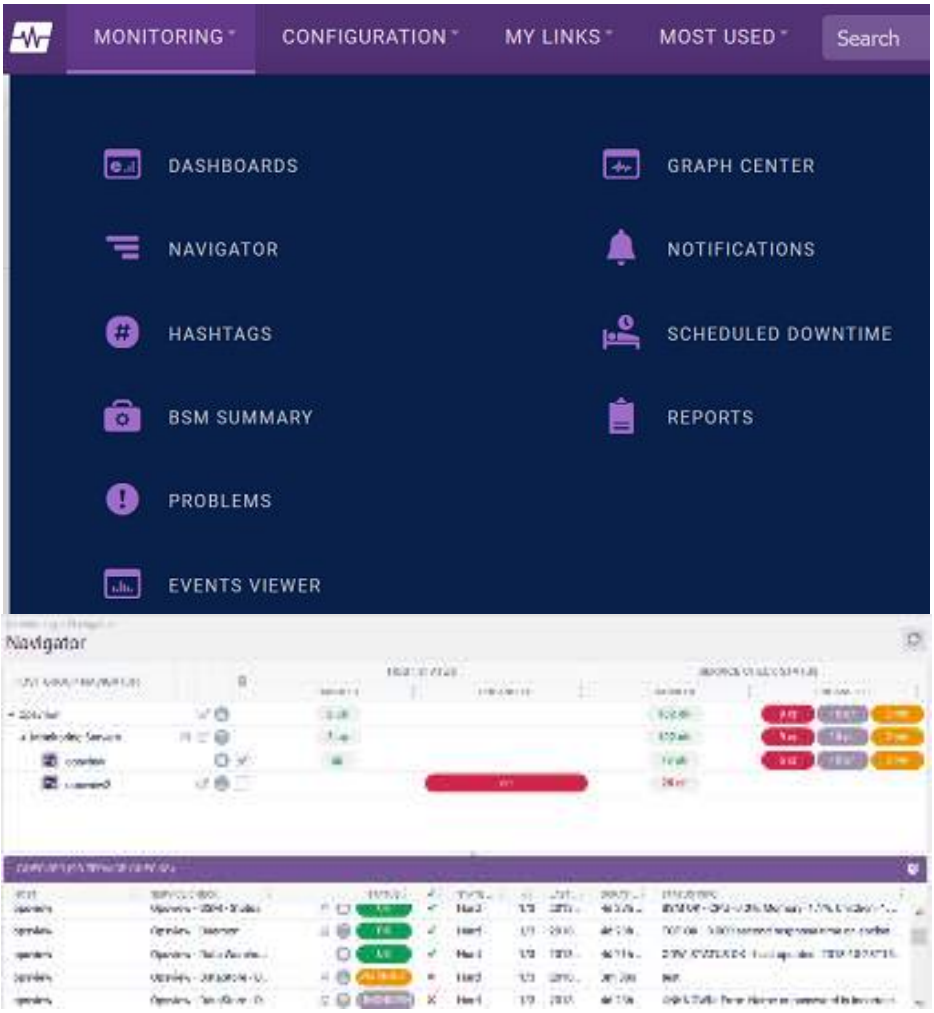

Рисунок 2.27 – Вид інтерфейсу Opsview

Переваги [64]:

– клонування сервісів;

– конфігурація;

– сповіщення.

Недоліки [64]:

– звітність;

– висока вартість.

Розглянуті програмні системи пропонують широкий спектр інструментів для моніторингу, аналізу та виявлення відхилень у роботі серверних систем. Деякі з них використовують методи машинного навчання для виявлення аномалій, аналізу часових рядів та визначення нетипових змін.

Для розробки системи виявлення аномального навантаження серверних систем важливими є такі функціональні можливості:

– моніторинг показників;
– періодичний збір і збереження показників;
– аналіз отриманих даних на основі порогових значень;
– застосування алгоритмів машинного навчання;
– порівняння результатів порогового аналізу та машинного навчання;
– збір лог-файлів;
– налаштування параметрів функціонування системи;
– формування сповіщень у разі перевантаження.

Розробка системи дозволяє:

– дослідити особливості використання алгоритму Isolation Forest;
– реалізувати збір та обробку показників;
– виконати аналіз отриманих показників;
– порівняти результати;
– забезпечити збереження даних і результатів виявлення аномалій;
– створити можливість налаштування параметрів моніторингу;
– виконати допоміжні функції;
– надіслати сповіщення на пошту;
– проінформувати користувача з допомогою інструкції.

У межах даної роботи було прийнято рішення про реалізацію системи виявлення аномального навантаження серверних систем на основі машинного навчання з використанням мови програмування Python, яка забезпечує необхідні засоби для збору показників, обробки даних, навчання моделі та аналізу отриманих результатів.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Під час розробки програмного забезпечення системи виявлення аномального навантаження серверних систем було розглянуто засоби та інструменти, які можуть використовуватися для реалізації моніторингу, аналізу даних і машинного навчання. Основними критеріями вибору стали простота реалізації, наявність бібліотек, підтримка алгоритмів машинного навчання, можливість збереження результатів та доступність документації.

Python – мова програмування високого рівня, яка застосовується для розробки ПЗ, аналізу даних та машинного навчання. Однією з її основних характеристик є зрозумілий синтаксис, що спрощує написання, перевірку та подальше вдосконалення програмного коду. Python підтримує модульну організацію програм, що дозволяє розділяти компоненти системи та забезпечувати їхню взаємодію.

Важливою особливістю Python є наявність розвиненої екосистеми бібліотек, призначених для математичних обчислень, статистичного аналізу, обробки інформації та реалізації алгоритмів машинного навчання. Це дозволяє використовувати програмні засоби для дослідження закономірностей та виявлення нетипових змін.

Python підтримує об’єктно-орієнтований, процедурний та функціональний підходи до програмування. Для розробки системи моніторингу серверних ресурсів важливими є можливості створення класів, функцій та окремих модулів, які відповідають за збір показників, аналіз навантаження та визначення відповідних аномальних станів [65].

Основні особливості Python:

- підтримка об’єктно-орієнтованого програмування;
- типізація даних;
- модульна організація програмного коду;
- підтримка поліморфізму, інкапсуляції та успадкування;

– можливість обробки винятків;
– робота з файлами;
– наявність вбудованих структур даних;
– підтримка сторонніх бібліотек;
– можливість виконання різних обчислень;
– метапрограмування;
– використання алгоритмів машинного навчання.

Модульна структура Python дозволяє розділяти програмне забезпечення на окремі компоненти, кожен із яких виконує завдання. Такий підхід є важливим для системи виявлення аномального навантаження, оскільки процес збору показників, їх аналіз та застосування алгоритму машинного навчання можуть реалізовуватися незалежними модулями.

Однією з характеристик Python є типізація, яка дозволяє працювати зі змінними без обов'язкового попереднього оголошення їхніх типів. Мова також підтримує обробку списків, словників та інших структур даних, що спрощує збереження й опрацювання показників використання серверних ресурсів.

Основні переваги мови програмування Python:
– простота синтаксису;
– відкритий код;
– підтримка різних ОС;
– можливість повторного використання програмних компонентів;
– наявність великої кількості бібліотек;
– зручність виконання різних операцій;
– підтримка роботи з файлами;
– доступність документації.

Основні недоліки мови програмування Python:
– низька швидкодія;
– додаткове використання оперативної пам'яті;
– обмеження підтримки платформ;

– залежність від версій бібліотек;

– проблеми сумісності.

Python є гарним вибором для розробки системи виявлення аномального навантаження серверних систем. Його можливості дозволяють реалізувати моніторинг ресурсів, обробку отриманих показників та використання алгоритмів машинного навчання [66].

Бібліотеки Python забезпечують виконання операцій, необхідних для збору, збереження та аналізу інформації. Для реалізації системи виявлення аномального навантаження особливе значення мають засоби роботи із системними показниками, лог-файлами, обчисленнями та алгоритмами машинного навчання.

Основні бібліотеки та модулі Python:

Math – модуль Python, який забезпечує доступ до математичних функцій та операцій.

Time – модуль, який дозволяє працювати з часовими інтервалами та встановлювати затримки між операціями.

Os – стандартний модуль для взаємодії з операційною системою, файловою структурою та параметрами.

Pathlib – модуль для роботи зі шляхами до файлів і каталогів, який забезпечує керування файловими ресурсами.

Datetime – модуль для роботи з датами та часом, який може використовуватися для формування часових позначок під час збору відповідних показників.

Json – модуль для зчитування та запису даних у форматі JSON, що може використовуватися для збереження результатів моніторингу та параметрів конфігурації.

Csv – модуль для читання та запису табличних даних у форматі CSV, який може застосовуватися для підготовки даних.

NumPy – бібліотека для роботи з масивами та виконання математичних операцій над даними.

Pandas – бібліотека для обробки та аналізу структурованих даних, яка забезпечує засоби фільтрації, групування та перетворення показників.

Matplotlib – бібліотека для побудови графіків і візуалізації числових даних, що може використовуватися для відображення змін навантаження серверних ресурсів.

Psutil – бібліотека для отримання інформації про використання системних ресурсів, таких як центральний процесор, оперативна пам’ять, дисковий простір та мережева активність.

Scikit-learn – бібліотека машинного навчання, яка містить алгоритми класифікації, регресії та виявлення аномалій. У межах даної роботи використовується алгоритм Isolation Forest.

Joblib – бібліотека для збереження та завантаження таких об’єктів Python, як навчені моделі машинного навчання, яка дозволяє використовувати їх під час аналізу нових даних.

SciPy – бібліотека для обчислень, яка містить інструменти аналізу, оптимізації та математичної обробки даних.

Flask – фреймворк, який дозволяє швидко створювати інтерфейс користувача.

Keras – API для побудови та навчання нейронних мереж.

Smtplib – стандартний модуль Python, який забезпечує надсилання повідомлень за допомогою протоколу SMTP.

Серед наведених бібліотек основними засобами реалізації розроблюваної системи є psutil, scikit-learn та joblib. Бібліотека psutil забезпечує отримання показників використання серверних ресурсів, scikit-learn дозволяє застосовувати алгоритм машинного навчання Isolation Forest, а joblib використовується для збереження моделі [66, 67].

Алгоритм Isolation Forest належить до методів машинного навчання без учителя та призначений для виявлення аномалій. Його робота зосереджується на побудові випадкових дерев ізоляції, за допомогою яких визначаються спостереження, що відрізняються від сукупності даних.

У межах розроблюваної системи для аналізу використовуються показники завантаження центрального процесора, оперативної пам’яті та дискового простору. Отримані значення застосовуються для навчання моделі та подальшого визначення аномальних станів системи.

Особливістю використання Isolation Forest є можливість аналізу показників без необхідності попереднього визначення порогів. Водночас пороговий метод дозволяє визначати перевантаження за встановленими значеннями використання ресурсів. Порівняння результатів цих підходів дає можливість дослідити відмінності між виявленням перевантажень та визначенням нетипових спостережень [68].

Основні можливості Python:

– автоматизація збору показників;
– обробка числових даних;
– виконання різних операцій;
– машинне навчання;
– збереження та завантаження моделей;
– робота з лог-файлами;
– налаштування параметрів ПЗ;
– штучний інтелект;
– реалізація модульної структури системи;
– обробка винятків;
– типізація;
– вбудовані структури даних;
– інтерактивний режим;
– асинхронне програмування.

Python широко застосовується в аналізі даних, автоматизації задач та машинному навчанні. Завдяки поєднанню зрозумілого синтаксису та бібліотек мова дозволяє реалізовувати програми та продовжує адаптуватися до сучасних потреб, що забезпечує її актуальність у галузі ІТ [66, 69].

Для створення системи виявлення аномального навантаження Python було обрано як основну мову програмування з таких причин:

1) Наявність бібліотек [67]:

– psutil - збір системних показників;

– flask - відтворення інтерфейсу;

– scikit-learn - реалізація алгоритму Isolation Forest;

– smtplib - відправка повідомлень;

– joblib - збереження та завантаження моделі.

2) Швидкість розробки та зрозумілий синтаксис – Python дозволяє створювати програмні модулі та моделі за короткий час.

3) Кросплатформеність – ПЗ може використовуватися в операційних системах Windows, Linux та macOS.

4) Доступність документації – документація Python і бібліотек містить матеріали щодо їх встановлення, налаштування та використання.

5) Інтеграція з компонентами та системами.

Вибір архітектури та технологій [67]:

1) Psutil – використовується для збору показників використання центрального процесора, оперативної пам’яті та дискового простору.

2) Flask – забезпечує реалізацію інтерфейсу та відображення результатів у реальному часі.

3) Threading, signal – застосовується для запуску модулів, з можливістю обробки сигналів зупинки.

4) Scikit-learn – забезпечує навчання моделі Isolation Forest і виявлення аномальних змін у показниках.

5) Joblib – використовується для збереження навченої моделі машинного навчання та її завантаження.

6) JSON – використовується для збереження показників моніторингу та параметрів конфігурації.

7) SMTP – використовується для надсилання повідомлень про перевищення встановлених порогових значень.

Середовище розробки:

1) Visual Studio Code – редактор програмного коду з підтримкою мови Python, засобами налагодження та можливістю використання додаткових розширень.

2) Python 3 – інтерпретатор мови програмування, який забезпечує виконання модулів та роботу бібліотек.

Вибір мови програмування Python та бібліотек обумовлений необхідністю реалізації збору показників, їх обробки та виявлення аномального навантаження за допомогою машинного навчання. Використання psutil, scikit-learn та joblib дозволяє створити систему для моніторингу серверних ресурсів та застосування алгоритму Isolation Forest.

2.3 Розгорнута постановка завдання

Метою роботи є дослідження та програмна реалізація системи виявлення аномального навантаження серверних систем на основі машинного навчання, яка забезпечує збір, обробку та аналіз показників використання серверних ресурсів, визначення перевантажень і нетипових станів. Розроблювана система повинна забезпечувати можливість спостереження за станом серверних ресурсів, збереження отриманих показників та використання алгоритму машинного навчання для виявлення аномального навантаження.

Основною особливістю системи є поєднання порогового аналізу з алгоритмом машинного навчання Isolation Forest. Пороговий метод дозволяє визначати перевантаження на основі попередньо встановлених значень використання ресурсів, тоді як Isolation Forest використовується для виявлення спостережень, які відрізняються від поведінки зібраних показників. Застосування двох підходів дозволяє дослідити їхні особливості та порівняти отримані результати.

Для досягнення поставленої мети необхідно реалізувати такі основні компоненти:

1) Модуль моніторингу – забезпечує збір показників використання центрального процесора, оперативної пам’яті та дискового простору. Отримані дані використовуються для подальшого аналізу навантаження.

2) Модуль аналізу даних – відповідає за обробку зібраної інформації та порівняння показників використання серверних ресурсів із встановленими пороговими значеннями. На основі отриманих результатів визначаються ознаки перевантаження. Пороговий аналіз також використовується для подальшого порівняння з результатами машинного навчання.

3) Модуль виявлення аномалій – призначений для виявлення аномального навантаження серверних систем із використанням алгоритму Isolation Forest. Модуль забезпечує зчитування показників моніторингу, підготовку даних для навчання моделі, визначення аномальних спостережень та збереження результатів аналізу. Для навчання використовуються показники завантаження процесора, оперативної пам’яті та дискового простору. Додатково передбачається порівняння результатів виявлення аномалій із результатами порогового аналізу для визначення випадків збігу та розбіжності між методами.

4) Модуль розподілу ресурсів – призначений для виконання допоміжних операцій, пов’язаних з аналізом використання серверних ресурсів та визначенням заходів щодо оптимізації навантаження.

5) Модуль сповіщення – забезпечує інформування про виявлення перевантажень серверних ресурсів. Формування повідомлень здійснюється на основі результатів аналізу показників та перевищення встановлених порогових значень. Це дозволяє своєчасно отримувати інформацію про можливі проблеми системи.

6) Модуль логування – забезпечує збереження показників моніторингу та результатів аналізу у відповідних лог-файлах. До збереженої інформації належать часові позначки, значення використання серверних ресурсів, результати виявлення аномалій та інші дані, необхідні для подальшого дослідження. Наявність лог-файлів дозволяє аналізувати зміни навантаження та перевіряти результати роботи ПЗ.

7) Модуль конфігурації – забезпечує налаштування основних параметрів функціонування системи, таких як інтервал збору показників та порогові значення використання серверних ресурсів. Використання конфігураційних параметрів дозволяє адаптувати роботу програмного забезпечення до заданих потреб користувача.

8) Модуль безпеки – передбачає використання допоміжних засобів контролю доступу до функцій системи. Його призначення полягає в обмеженні несанкціонованого використання програмного забезпечення. При цьому контроль доступу базується на перевірці терміну дії токенів автентифікації, а всі події успішних чи невдалих входів фіксуються у лог-файлі.

9) Модуль тестування – передбачає перевірку правильності функціонування основних програмних модулів. Тестування охоплює збір показників серверних ресурсів, виконання порогового аналізу, навчання самої моделі, виявлення аномальних спостережень та збереження результатів. Це дозволяє оцінити коректність функцій і виявити можливі помилки під час роботи системи.

10) Модуль документації – містить опис структури програмного забезпечення, призначення його основних компонентів та порядок використання системи. Документація забезпечує користувача необхідною інформацією щодо запуску моніторингу, налаштування параметрів, переходу за посиланнями, виконання аналізу та перегляду результатів виявлення аномального навантаження.

Реалізація зазначених модулів дозволить створити програмне забезпечення системи виявлення аномального навантаження серверних систем на основі машинного навчання. Основним напрямом дослідження є застосування алгоритму Isolation Forest до показників використання серверних ресурсів і порівняння його результатів із пороговим методом аналізу. Поєднання таких підходів забезпечить можливість дослідження особливостей виявлення перевантажень та аномальних станів серверних систем.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Розроблювана система виявлення аномального навантаження серверних систем на основі машинного навчання призначена для моніторингу використання серверних ресурсів, аналізу показників та визначення нетипових станів. Її функціонування базується на взаємодії модулів, які забезпечують збір даних, пороговий аналіз, застосування алгоритму Isolation Forest та збереження результатів.

Основні етапи функціонування системи:

1) Конфігурація:

Після запуску система отримує основні налаштування з конфігураційного файлу, який містить параметри:

- інтервал;
- порогові значення.

Отримані параметри використовуються для налаштування періодичності моніторингу та визначення перевантажень. Також їх можна змінювати у самому файлі.

2) Моніторинг:

Модуль моніторингу отримує показники:

- рівень завантаження центрального процесора;
- використання оперативної пам’яті;
- використання дискового простору;
- вхідний мережевий трафік;
- вихідний мережевий трафік.

Для отримання інформації про стан ресурсів використовується бібліотека psutil. Зібрані показники зберігаються у лог-файлі та використовуються для подальшого аналізу.

3) Аналіз даних:

Модуль аналізу виконує порівняння отриманих показників із пороговими значеннями. У разі перевищення встановлених меж визначається стан перевантаження.

Результат порогового аналізу містить:

– стан перевантаження;

– перелік показників;

– критичний стан.

Отримані результати використовуються для формування сповіщень про перевантаження або критичний стан та подальшого порівняння з результатами машинного навчання.

4) Виявлення аномального навантаження:

Аномальне навантаження серверної системи означає нетиповий стан її роботи, що виникає при різких відхиленнях у сукупності показників використання центрального процесора, оперативної пам’яті та дискового простору порівняно зі звичайним режимом функціонування самої системи.

Основною складовою розробленої системи є модуль машинного навчання, призначений для виявлення нетипових змін у показниках використання серверних ресурсів. Для його реалізації використано алгоритм Isolation Forest.

Математична суть алгоритму Isolation Forest полягає в побудові набору випадкових дерев рішень, які послідовно розділяють зібрані показники навантаження. Оскільки аномальні значення трапляються рідко і суттєво відрізняються від звичайної роботи сервера, алгоритм відокремлює їх значно швидше. На основі середньої кількості таких кроків система оцінює кожен запис: якщо він ізолюється швидко, алгоритм позначає його як аномалію, а якщо потребує багато кроків – як нормальний стан системи [2].

Вхідними даними для модуля є показники використання центрального процесора, оперативної пам’яті та дискового простору, отримані модулем моніторингу. Ці дані зчитуються з лог-файлу та формуються у набір ознак, який використовується для навчання й подальшого аналізу.

На початковому етапі роботи модуль перевіряє наявність достатньої кількості зібраних записів для навчання моделі. Після цього створюється модель Isolation Forest із заданими параметрами та виконується її навчання на підготовлених показниках. Навчена модель зберігається у файлі для подальшого використання, що дозволяє не виконувати навчання повторно під час кожного запуску програми.

Під час наступного аналізу завантажується збережена модель і перевіряються нові записи моніторингу. Алгоритм визначає, чи відповідають отримані показники нормальній поведінці системи. Результат класифікації дає змогу встановити, чи є конкретне спостереження аномальним. Виявлені відхилення та результати аналізу записуються до окремого лог-файлу.

Основні операції:

– зчитування та підготовка показників моніторингу;

– навчання моделі та збереження моделі;

– виявлення аномального навантаження серверної системи;

– порівняння результатів із пороговим аналізом та запис результатів.

5) Розподіл ресурсів:

Модуль розподілу ресурсів використовується для виконання допоміжних операцій, спрямованих на зменшення навантаження серверної системи.

6) Система сповіщень:

У разі виявлення перевищення або аномального стану модуль сповіщення формує повідомлення з інформацією про стан ресурсів.

7) Логування:

Під час функціонування системи здійснюється збереження показників моніторингу та результатів їх аналізу у лог-файл.

8) Безпека та доступ:

Для обмеження доступу є допоміжні функції автентифікації та авторизації.

9) Тестування та документація:

Модулі тестування та документації забезпечують перевірку функцій ПЗ та надання користувачу інформації щодо роботи системи.

3.2 Розробка структурної схеми

Структурна схема системи виявлення аномального навантаження серверних систем на основі машинного навчання відображає основні програмні компоненти та взаємозв’язки між ними. Вона дозволяє визначити загальну структуру програмного забезпечення, призначення модулів та порядок обміну даними між ними [70, 71].

Система побудована за модульним принципом, де кожен компонент виконує окремі функції. Взаємодія між модулями здійснюється шляхом передавання показників моніторингу, результатів аналізу та використання спільних файлів даних. Особливістю структури є поєднання порогового методу аналізу з алгоритмом машинного навчання Isolation Forest.

Основні компоненти структурної схеми:

1) Користувач:

– здійснює взаємодію із системою;

– налаштовує параметри моніторингу;

– переглядає результати аналізу та лог-файлів.

2) Модуль конфігурації:

– зберігає налаштування;

– дозволяє змінювати параметри.

3) Модуль моніторингу:

– здійснює збір показників;

– забезпечує передавання отриманих даних для подальшого аналізу;

– зберігає показники у лог-файлі.

4) Модуль аналізу даних:

– оцінює отримані показники;

– порівнює їх із пороговими значеннями;

– визначає перевантаження;

– формує результати для подальшого порівняння.

5) Модуль виявлення аномалій:

– зчитує збережені показники моніторингу;

– здійснює підготовку даних;

– використовує алгоритм Isolation Forest;

– забезпечує збереження та завантаження моделі;

– порівнює результати;

– записує результати виявлення аномалій у лог-файл.

6) Модуль розподілу ресурсів:

– виконує операції з управління серверними ресурсами;

– забезпечує виконання допоміжних дій;

– фіксує результати виконаних дій.

7) Модуль сповіщення:

– отримує інформацію про перевищення порогових значень;

– формує повідомлення;

– забезпечує надсилання сповіщення.

8) Модуль логування:

– забезпечує збереження показників;

– записує результати;

– надає дані для подальшого дослідження.

9) Модуль безпеки:

– забезпечує контроль доступу;

– виконує перевірку даних;

– записує результати.

10) Модуль тестування:

– забезпечує перевірку правильності функціонування системи.

– сприяє подальшому вдосконаленню ПЗ.

11) Модуль документації:

– містить інструкцію щодо налаштування та використання системи.

Вигляд структурної схеми:

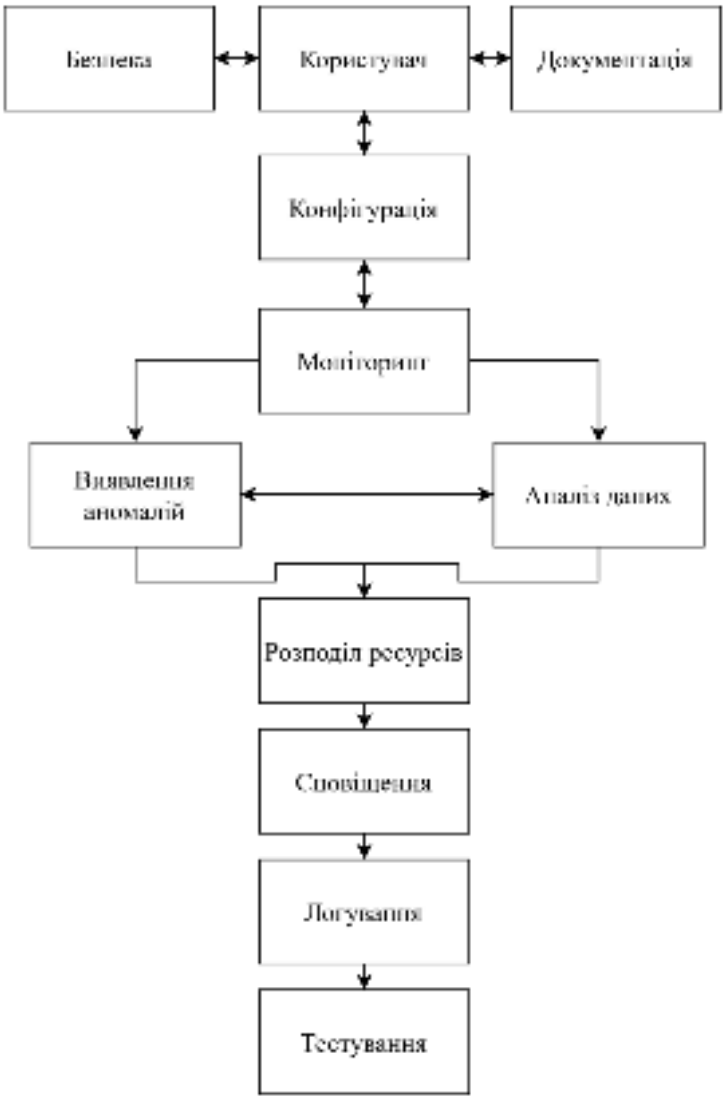

Рисунок 3.1 – Структурна схема

Структурна схема відображає взаємозв’язок основних компонентів системи, кожен із яких виконує функції та має чітке призначення. Взаємодія модулів забезпечує виконання моніторингу, аналізу показників, виявлення аномального навантаження та збереження результатів.

3.3 Розробка функціональної схеми

Функціональна схема системи відображає послідовність виконання операцій та взаємодію програмних модулів, які забезпечують роботу самої системи та зручність використання для користувача [72].

Основні функціональні модулі:

1) Модуль безпеки:

Відповідає за перевірку доступу та захист даних.

2) Модуль конфігурації:

Забезпечує налаштування основних параметрів роботи системи: інтервалу збору показників та порогових значень. Працює з конфігураційним файлом, із якого отримуються параметри. Підтримує динамічне оновлення через інтерфейс із подальшою передачею до модулів системи.

3) Модуль моніторингу:

Відповідає за збирання показників використання центрального процесора, оперативної пам’яті, дискового простору та обсягів мережевого трафіку за допомогою бібліотеки psutil. Зібрані показники передаються для подальшого аналізу та зберігаються у лог-файлі.

4) Модуль аналізу даних:

Виконує перевірку отриманих показників. У разі перевищення порогів визначається стан перевантаження.

5) Модуль виявлення аномалій (Ізолюючий ліс):

Здійснює аналіз зібраних показників із використанням алгоритму Isolation Forest. На основі даних виконується навчання моделі машинного навчання, після чого модель зберігається для подальшого використання. Інформація про виявлені аномалії, перевантаження та випадки збігу між методами записується у лог-файл.

6) Модуль розподілу ресурсів:

Призначений для виконання допоміжних операцій. За наявності відповідних функцій може виконувати дії для зменшення навантаження серверних систем.

7) Модуль сповіщення:

Забезпечує інформування про перевищення встановлених порогових значень або виявлення аномального стану. На основі результатів формуються повідомлення про виявлені відхилення із зазначенням метрик, які можуть передаватися електронною поштою.

8) Модуль логування:

Здійснює збереження показників. Інформація записується у відповідні лог-файли, що дозволяє використовувати її для подальшого аналізу.

Вигляд функціональної схеми:



Рисунок 3.2 – Функціональна схема

Функціональна схема відображає логіку роботи системи та взаємодію між її основними модулями. Взаємозв’язок модулів забезпечує функціонування всієї програми. Кожен з них відповідає за свою функцію.

3.4 Розробка діаграми процесів

Діаграма процесів системи відображає послідовність виконання основних операцій під час моніторингу серверних ресурсів, обробки показників та визначення аномальних станів. Вона дозволяє відслідковувати порядок взаємодії модулів, дій користувача та переходи між етапами функціонування всієї системи.

Використання діаграми процесів дозволяє відобразити логіку збору та аналізу даних, визначити взаємозв'язки між пороговим методом і алгоритмом машинного навчання Isolation Forest. Крім того, діаграма спрощує перевірку послідовності виконання операцій, виявлення можливих помилок та подальше вдосконалення програмного забезпечення. Вона також може використовуватися під час тестування, документування та дослідження особливостей роботи системи [73].

Основні процеси в системі:

1) Запуск системи – здійснюється ініціалізація модулів, завантаження конфігураційних параметрів та встановлення інтервалу збору показників.

2) Моніторинг показників – зчитування та збір даних про завантаження центрального процесора, використання оперативної пам'яті та дискового простору.

3) Навчання моделі – підготовка збережених показників для використання алгоритмом Isolation Forest.

4) Аналіз даних – порівняння показників із пороговими значеннями.

5) Виявлення аномалій – використання навченої моделі Isolation Forest для подальшого аналізу показників.

6) Рішення та порівняння результатів:

– якщо немає навантаження - запис у відповідний лог-файл;

– якщо є перевантаження - активація модулів для його усунення.

– збіг результатів - використання для подальшого аналізу.

7) Розподіл ресурсів та логування – виконання допоміжних операцій та фіксація дій.

8) Забезпечення безпеки – контроль доступу користувача та захист даних.

9) Тестування та документація – перевірка правильності функціонування модулів та надання інструкції.

Вигляд діаграми процесів:

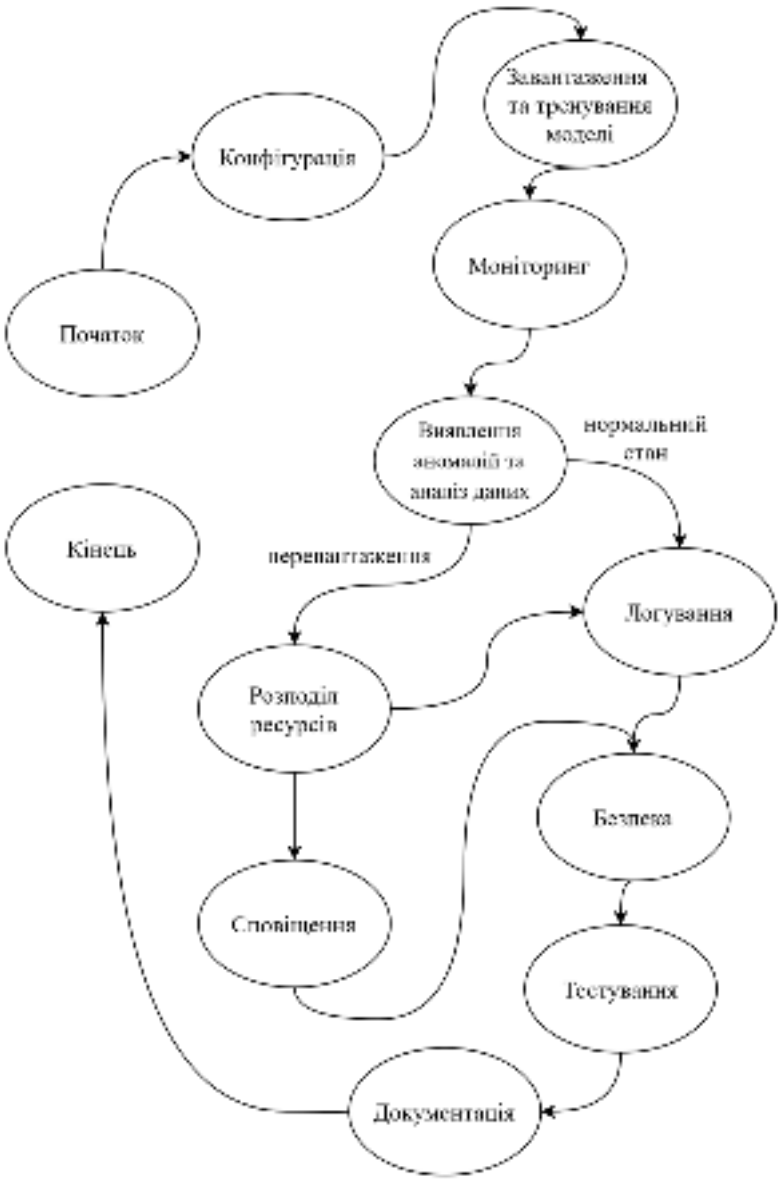

Рисунок 3.3 – Діаграма процесів

Діаграма процесів відображає послідовність виконання основних операцій та ключові моменти. Представлення процесів у вигляді діаграми сприяє структуруванню всієї системи та спрощує її подальше тестування й удосконалення.

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

4.1 Розробка блок-схем та опис алгоритмів функціонування системи

Система виявлення аномального навантаження серверних систем на основі машинного навчання реалізується мовою програмування Python. Кожен модуль виконує окремі функції. Такий підхід дозволяє розділити основні процеси системи, спростити перевірку правильності роботи модулів та забезпечити можливість їх подальшого вдосконалення.

Блок-схема основної програми:

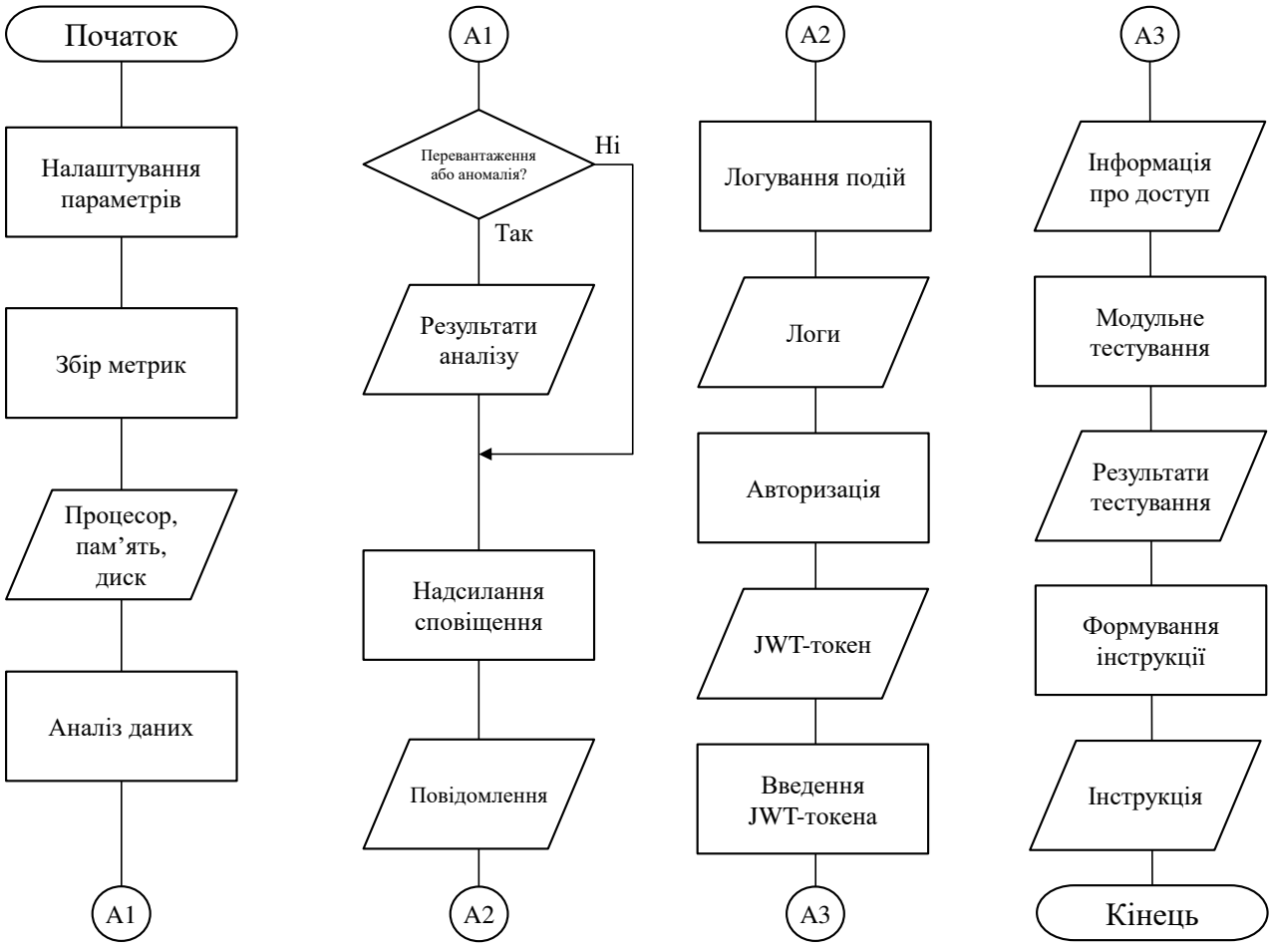

Рисунок 4.1 – Блок-схема основної програми

Блок-схема роботи підпрограми:

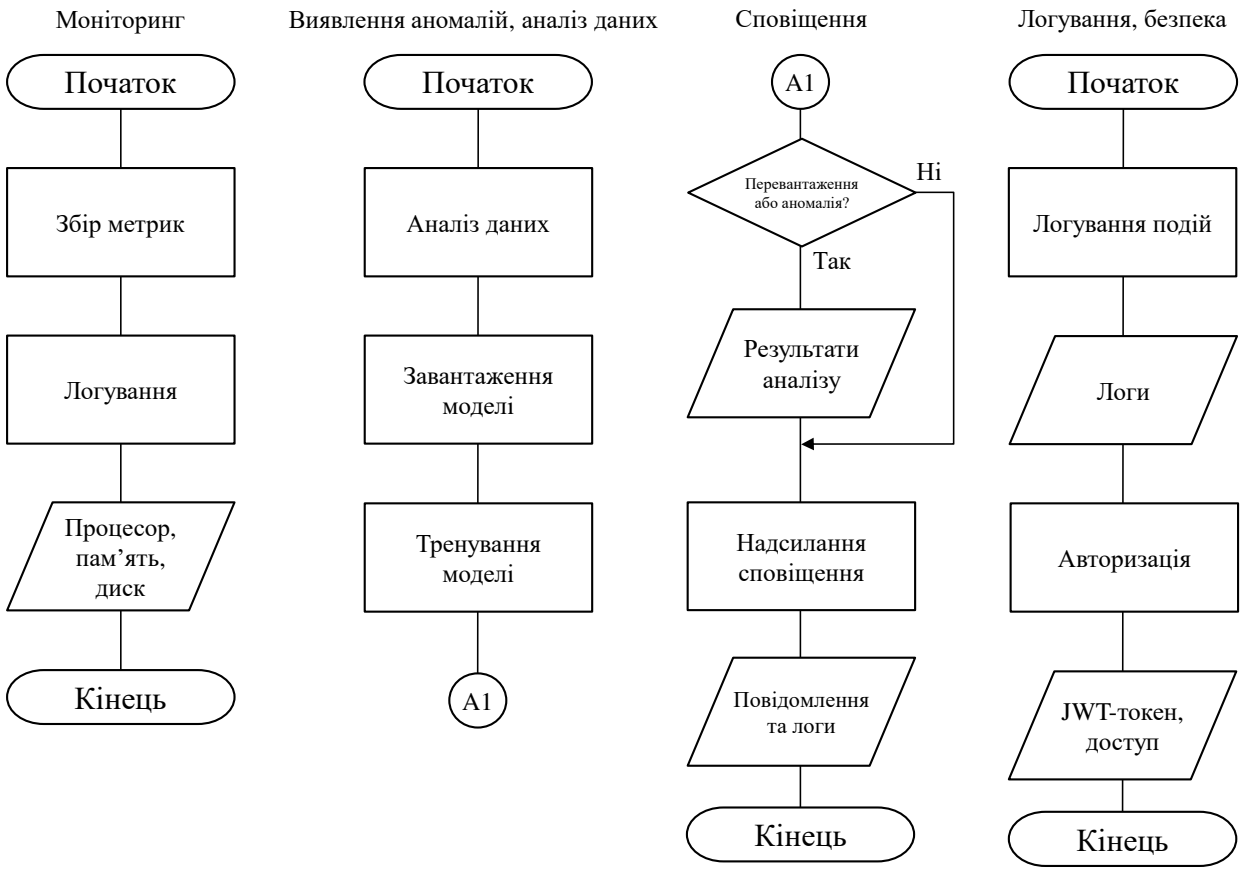

Рисунок 4.2 – Блок-схема роботи підпрограми

На рисунку 4.2 наведено блок-схему роботи підпрограми, розділену на три гілки: моніторингу; виявлення аномалій, аналізу даних, сповіщення; логування та безпеки.

Підпрограма моніторингу: запускає процес збору метрик, записує їх у лог-файл та виводить показники.

Підпрограма виявлення аномалій, аналізу даних та сповіщення: виконує пороговий аналіз даних, завантажує та тренує модель машинного навчання, після чого перевіряє умову наявності перевантаження або аномалії. У разі виявлення відхилення система формує результати аналізу, надсилає сповіщення та фіксує отриману інформацію у лог-файлах.

Підпрограма логування та безпеки: відповідає за фіксацію подій у лог-файли та виконання процедури авторизації користувача для надання доступу.

Призначення даної блок-схеми полягає у відображенні роботи окремих модулів, щоб не ускладнювати загальну схему основної програми. Це дає змогу більш детально показати роботу кожної з підпрограм.

Алгоритм конфігурації:

Забезпечує ініціалізацію та налаштування параметрів роботи всієї системи. На етапі виконання програма зчитує конфігураційний файл, де задаються інтервал та порогові значення. На блок-схемі основної програми алгоритм реалізовано блоком процесу «Налаштування параметрів».

Алгоритм моніторингу:

Призначений для збирання метрик. За допомогою бібліотеки psutil алгоритм опитує стан операційної системи з інтервалом, визначає відсоток завантаження центрального процесора, обсяг оперативної пам’яті та дискового простору. На блок-схемі основної програми цей алгоритм представлено блоком процесу «Збір метрик» та блоком виводу «Процесор, пам’ять, диск».

Алгоритм аналізу даних:

Виконує перевірку зібраних показників. Алгоритм отримує значення метрик від модуля моніторингу та зіставляє із відповідним порогом. Якщо хоча б один параметр перевищує допустиму межу, алгоритм визначає стан перевантаження. На блок-схемі основної програми алгоритм відображено блоком процесу «Аналіз даних» та в умовному блоці «Перевантаження або аномалія?».

Алгоритм виявлення аномалій:

Реалізує аналіз навантаження на основі методу машинного навчання Isolation Forest. Спочатку алгоритм зчитує показники з лог-файлу. Виконується завантаження моделі та її тренування. Далі алгоритм визначає наявність аномалій, зіставляє їх із результатами порогового аналізу та записує у лог-файл. На блок-схемі роботи підпрограм алгоритм показано блоками процесу «Завантаження моделі» та «Тренування моделі», умовним блоком «Перевантаження або аномалія?», блоком виводу «Результати аналізу». На блок-схемі основної програми він інтегрований в блок процесу «Аналіз даних», умовний блок «Перевантаження або аномалія?», блок виводу «Результати аналізу».

Алгоритм сповіщення:

Відповідає за інформування про виявлені стани перевантаження. Алгоритм активується лише тоді, коли пороговий аналіз або модель Isolation Forest визначають відхилення від норми. На основі отриманих даних формується повідомлення та надсилається на пошту через протокол SMTP. На блок-схемі основної програми алгоритм представлений блоком процесу «Надсилання сповіщення» та блоком виводу «Повідомлення».

Алгоритм логування:

Забезпечує фіксацію подій у лог-файли. На блок-схемі основної програми алгоритм розміщено у вигляді блоку процесу «Логування подій» та блоку виводу «Логи».

Алгоритм безпеки:

Перевіряє дані користувача під час входу, генерує JWT-токен та перевіряє його для надання доступу. На блок-схемі основної програми алгоритм розміщено у вигляді блоків процесу «Авторизація», «Введення JWT-токена», блоків виводу «JWT-токен» та «Інформація про доступ».

Алгоритм тестування:

Призначений для перевірки правильності роботи модулів. На блок-схемі основної програми алгоритм відображено у вигляді блоку процесу «Модульне тестування» та блоку виводу «Результати тестування».

Алгоритм документації:

Формує інструкцію користувача. На блок-схемі основної програми алгоритм представлений блоком процесу «Формування інструкції» та блоком виводу «Інструкція».

Розроблені блок-схеми та алгоритми описують основні процеси функціонування системи виявлення аномального навантаження серверних систем. Кожен модуль виконує окреме завдання та взаємодіє з іншими компонентами. Особливу роль відіграє алгоритм Isolation Forest, який дозволяє визначати аномалії. Опис алгоритмів створює основу для проведення подальших досліджень.

Збір показників стану серверної системи реалізовано в модулі mon.py. Для отримання інформації про використання ресурсів застосовується бібліотека psutil, яка надає можливість отримувати поточні характеристики процесора, оперативної пам’яті, дискового простору та мережевої активності.

Основною функцією збору даних є col_metr(), яка отримує метрики та повертає їх у вигляді словника. Формат виклику такої функції наведено в наступному фрагменті:

```
def col_metr(s):
    metrics = {
        'datetime': time.strftime('%Y-%m-%d %H:%M:%S'),
        'cpu_usage': psutil.cpu_percent(interval=1),
        'memory_usage': psutil.virtual_memory().percent,
        'disk_usage': psutil.disk_usage('/').percent,
        'network': {
            'bytes_sent': psutil.net_io_counters().bytes_sent,
            'bytes_recv': psutil.net_io_counters().bytes_recv
        }
    }
    return metrics
```

У процесі виконання функції формується словник metrics, який містить дату й час отримання показників, відсоток використання центрального процесора, оперативної пам’яті та дискового простору, а також вхідний і вихідний мережевий трафік.

Для отримання відповідних характеристик використовуються функції psutil.cpu_percent(), psutil.virtual_memory(), psutil.disk_usage() та psutil.net_io_counters(). Функція time.strftime() забезпечує формування часової позначки вимірювання.

Збереження отриманих показників виконується методом log_metr(), який записує словник до файлу server_metrics.log у форматі JSON. Для цього застосовується функція json.dumps(), яка перетворює об’єкт Python на текстове JSON-представлення. Використання формату JSON робить його зручним для перегляду.

Для визначення перевантаження використовується модуль ad.py, у якому реалізовано клас Dataanalysis та функцію anldt(). Призначення даної функції полягає у перевірці отриманих показників на відповідність заданим пороговим значенням. Формат виклику функції anldt():

```
    def anldt(s, metrics):
        iss = {
            'overload': False,
            'critical': False,
            'desc': []
        }
        if metrics['cpu_usage'] >= s.thresholds['cpu']:
            iss['overload'] = True
            iss['desc'].append("High cpu usage: " + str(metrics['cpu_usage']) +
"%")
        if metrics['memory_usage'] >= s.thresholds['memory']:
            iss['overload'] = True
            iss['desc'].append("High memory usage: " +
str(metrics['memory_usage']) + "%")
        if metrics['disk_usage'] >= s.thresholds['disk']:
            iss['overload'] = True
            iss['desc'].append("High disk usage: " + str(metrics['disk_usage']) +
"%")
        if metrics['cpu_usage'] >= s.thresholds['cpu'] and \
           metrics['memory_usage'] >= s.thresholds['memory'] and \
           metrics['disk_usage'] >= s.thresholds['disk']:
            iss['critical'] = True
        return iss
```

Функція anldt() приймає словник metrics, отриманий від модуля моніторингу. Для кожного показника виконується порівняння з відповідним порогом, який зберігається в конфігурації системи.

У результаті формується словник iss, що містить три основні поля: overload визначає наявність перевантаження, critical вказує на критичний стан, а desc містить опис виявлених перевищень. Критичний стан встановлюється у випадку одночасного перевищення порогів усіх трьох ресурсів. Реалізація порогового аналізу дає змогу своєчасно визначати перевантаження на основі встановлених обмежень.

Основним компонентом є модуль aml.py. Він призначений для виявлення нетипових станів серверної системи за допомогою алгоритму Isolation Forest, реалізованого в бібліотеці scikit-learn, яка є важливою для машинного навчання і забезпечує високу швидкість обробки даних.

Для роботи модуля використовуються показники завантаження центрального процесора, оперативної пам’яті та дискового простору. На відміну від порогового методу, який перевіряє окремі значення за встановленими обмеженнями, Isolation Forest аналізує сукупність показників і визначає спостереження, які відрізняються від характерної для навчальних даних поведінки.

Підготовка даних для навчання моделі виконується функцією l_metr(). Вона зчитує записи з файлу server_metrics.log, отримує часові позначки та формує набір числових характеристик. Формат виклику такої функції наведено в наступному фрагменті:

```
def l_metr():
    data = []
    dates = []
    with open("server_metrics.log", "r", encoding="utf-8") as file:
        for line in file:
            if not line.strip():
                continue
            metrics = json.loads(line)
            dates.append(metrics["datetime"])
            data.append([
                metrics["cpu_usage"],
                metrics["memory_usage"],
                metrics["disk_usage"]
            ])
    return data, dates
```

Для читання записів використовується функція json.loads(), яка перетворює JSON-рядок на об’єкт Python. Отримані числові показники додаються до відповідного списку data, а відповідні часові позначки – до списку dates. У результаті функція повертає два набори даних, необхідні для подальшого навчання та аналізу.

Навчання моделі реалізовано у функції tr_mdl(). Перед початком навчання перевіряється кількість доступних записів. Якщо їх менше десяти, процес навчання не виконується, оскільки для початкової реалізації передбачено мінімальний обсяг даних.

Основний фрагмент створення та навчання відповідної моделі має такий вигляд:

```
    mdl = IsolationForest(
        contamination=0.1,
        random_state=42
    )
    mdl.fit(data)
```

Клас IsolationForest використовується для створення моделі виявлення аномалій. Параметр contamination задає відсоток аномальних спостережень у наборі даних, а random_state забезпечує стабільність результатів випадкових операцій алгоритму. Метод fit() виконує навчання створеної моделі на підготовлених даних.

Після завершення навчання модель зберігається за допомогою бібліотеки joblib. Для цього використовується функція joblib.dump(), яка записує об’єкт моделі та кількість записів до файлу an_mod.pkl. Фрагмент збереження навченої моделі:

```
    joblib.dump({
        "model": mdl,
        "proc_count": len(data)
    }, "an_mod.pkl")
```

Збереження кількості записів дає змогу ефективно відокремлювати нові вимірювання від тих, які вже використовувалися під час навчання або попереднього аналізу.

Виявлення аномалій у нових записах виконується функцією det_anom(). На початку її роботи за допомогою joblib.load() завантажується раніше навчена модель та інформація про кількість записів. Після цього з лог-файлу моніторингу отримуються дані.

Для визначення аномальних спостережень застосовується метод predict():

```
    res = mdl.predict(n_dt)
```

Для додаткової оцінки результатів у функції det_anom() виконується порівняння висновків Isolation Forest із результатами порогового аналізу. Для кожного нового запису формується набір показників, після чого викликається метод anldt().

Під час порівняння визначається кількість аномалій, виявлених алгоритмом машинного навчання Isolation Forest, кількість перевантажень, установлених пороговим методом, а також кількість збігів. Отримана інформація виводиться в консоль і записується до anom_res.log.

Додатково в модулі реалізовано функцію chk_an(), яка дає змогу перевіряти та виявляти аномальне навантаження серверної системи за допомогою вже навченої моделі. Формат виклику такої функції наведено в наступному фрагменті:

```
def chk_an(metrics):
    try:
        saved = joblib.load("an_mod.pkl")
        mdl = saved["model"]
        data = [[
            metrics["cpu_usage"],
            metrics["memory_usage"],
```

```
            metrics["disk_usage"]
        ]]
        return mdl.predict(data)[0] == -1
    except FileNotFoundError:
        print("Model not found. Train first")
        return False
```

Функція приймає словник метрик, формує з нього набір числових показників і передає його моделі для перевірки. Якщо результат прогнозування дорівнює -1, функція повертає значення True, що свідчить про аномальне навантаження серверної системи, а в іншому випадку повертається False. При відсутності файлу моделі виводиться відповідне повідомлення.

Модуль aml.py забезпечує цикл застосування машинного навчання: підготовку даних, навчання моделі, збереження її стану, аналіз нових вимірювань та порівняння отриманих результатів із пороговим методом.

Для виконання допоміжних операцій використовується модуль ar.py, у якому реалізовано клас Resourceallocation. Його функція re_res() приймає поточні метрики системи та перевіряє їх відповідність до встановлених порогових значень.

Залежно від результатів перевірки викликаються методи: cpu_us() для обробки процесів, mem_us() для роботи з оперативною пам’яттю та disk_us() для очищення тимчасових файлів.

Для отримання інформації про процеси застосовується бібліотека psutil. Метод re_res() формує список дій, який повертається після завершення операцій. Цей компонент має допоміжне призначення.

Надсилання повідомлень реалізовано в модулі no.py. Для формування та передавання електронних листів використовуються бібліотеки smtplib, email.mime.text та email.mime.multipart.

Основною функцією є send_emnot(), яка приймає тему повідомлення, текст і список отримувачів. Для створення листа формується об’єкт MIMEMultipart, до якого додається текст за допомогою MIMEText. Формат виклику функції send_emnot():

```
def send_emnot(subject, message, rec):
    try:
        msg = MIMEMultipart()
        msg['From'] = G_EMAIL
```

```
        msg['To'] = ', '.join(rec)
        msg['Subject'] = subject
        msg.attach(MIMEText(message, 'plain'))
        server = smtplib.SMTP('smtp.gmail.com', 587)
        server.starttls()
        server.login(G_EMAIL, G_PASSWORD)
        server.sendmail(G_EMAIL, rec, msg.as_string())
        server.quit()
        print(f"Notification sent: {subject}")
    except Exception as e:
        print(f"Failed to send notification: {e}")
```

Передавання повідомлення здійснюється через SMTP-сервер із використанням захищеного з'єднання TLS. Після успішного надсилання в консоль виводиться повідомлення про результат. У разі виникнення помилки передбачено виведення відповідного повідомлення та обробку винятку.

У функції ano() виконується взаємодія модулів моніторингу, порогового аналізу, управління ресурсами та сповіщення. Якщо пороговий аналіз визначає перевантаження, формується повідомлення. Додатково викликається chk_an(metrics) з модуля aml.py, що дозволяє перевірити показники за допомогою моделі Isolation Forest.

Модуль сповіщення забезпечує інформування про перевантаження та дає змогу використовувати результат перевірки машинним навчанням.

Налаштування параметрів системи реалізовано в модулі cn.py. Для роботи з конфігураційними даними використовується стандартна бібліотека json, яка забезпечує зчитування та збереження інформації у файлі config.json.

У модулі створено клас Configuration, методами якого є l_conf(), g_conf() та u_conf(). Перший метод завантажує конфігураційні дані з файлу, другий повертає налаштування, а третій забезпечує їх оновлення та запис.

Фрагмент отримання налаштувань:

```
def g_conf(s):
    return s.config
```

Поточна конфігурація містить порогові значення для центрального процесора, оперативної пам'яті та дискового простору, а також інтервал. Для зміни параметрів використовується форма, з якої введені значення отримуються та передаються до функції u_conf(). Формат виклику такої функції наведено в наступному фрагменті:

```
def u_conf(s, new_con):
```

```
        try:
            with open(F_CON, 'w', encoding='utf-8') as f:
                json.dump(new_con, f, indent=4)
            s.config = new_con
            logging.info("Configuration updated")
        except Exception as e:
            logging.error(f"Error: {e}")
```

Для реалізації інтерфейсу використовується бібліотека flask. Функція render_template_string() використовується для формування HTML-сторінки з налаштуваннями. Модуль конфігурації забезпечує зміну параметрів без необхідності редагування програмного коду.

Для фіксації подій та результатів роботи системи використовується модуль lg.py. Основою його реалізації є стандартна бібліотека logging, яка забезпечує запис повідомлень, попереджень і помилок.

Для формування заданих записів передбачено функцію log_ev(st, message). Вона дозволяє фіксувати важливі етапи роботи програми. Записи зберігаються у файлі system_logs.log. Формат виклику такої функції наведено в наступному фрагменті:

```
def log_ev(st, message):
    if st == 'info':
        logging.info(message)
    elif st == 'warning':
        logging.warning(message)
    elif st == 'error':
        logging.error(message)
    print(message)
```

Додатково модуль використовує бібліотеку flask для відображення інтерфейсу. Функція disp_logs() відображає сторінку лог-файлу, g_logs() повертає його вміст, а c_logs() забезпечує очищення. Для передавання даних у форматі JSON використовується функція jsonify(). Фрагмент виклику функції disp_logs():

```
def disp_logs():
    try:
        with open('system_logs.log', 'r') as f:
            logs = f.readlines()
    except FileNotFoundError:
        logs = ['No logs']
    html_temp = """…"""
    return render_template_string(html_temp, logs=''.join(logs))
```

Модуль логування дає змогу зберігати інформацію про роботу системи, результати моніторингу, виявлені перевантаження та різні операції, які можуть використовуватися для подальшого вдосконалення.

Для реалізації механізму авторизації користувачів використовується модуль cr.py. Він забезпечує перевірку даних, формування токенів та контроль доступу.

Для хешування паролів застосовується стандартна бібліотека hashlib, а для створення та перевірки токенів – бібліотека jwt. У модулі реалізовано функції generate_jwt() та verify_jwt(). Перша формує JWT-токен із зазначенням користувача та терміну його дії, друга перевіряє достовірність токена й повертає ім’я користувача або виводить відповідне повідомлення про помилку.

Функція login() забезпечує отримання даних та їх перевірку, а protected() використовується для перевірки токена. Інформація про спроби авторизації записується до файлу security_logs.log.

Для перевірки правильності функціонування програмних компонентів використовується модуль tt.py. Тестування реалізовано за допомогою бібліотеки unittest, яка дає змогу створювати та виконувати перевірки роботи окремих модулів.

У класі Testmodules передбачено перевірку збору метрик, правильності роботи порогового аналізу, формування та перевірки JWT-токенів, а також виклику функції chk_an() для перевірки аномального навантаження. Формат виклику функції test_aml():

```
def test_aml(s):
    t_dt = {
        'cpu_usage': 90,
        'memory_usage': 95,
        'disk_usage': 90
    }
    result = chk_an(t_dt)
    s.assertIn(result, [True, False])
```

Для перевірки результатів застосовуються методи assertIsInstance(), assertIn(), assertTrue() та assertEqual().

Для надання користувачеві інформації щодо порядку запуску та використання програмних компонентів системи передбачено модуль dc.py. У ньому реалізовано функцію dp_ins(), яка виводить інструкцію з використання системи, описує призначення основних модулів і правильну послідовність їх запуску.

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблена система виявлення аномального навантаження серверних систем призначена для використання на підприємствах, де необхідно контролювати стан ресурсів та виявляти відхилення від звичайного режиму роботи. Завдяки модульній структурі ПЗ може бути адаптоване до різних сфер використання та налаштоване відповідно до потреб користувача.

Основні вимоги:

1) Операційна система: Windows, Linux.

2) Мова програмування: Python.

3) Середовище розробки: VSC або інше.

4) Бібліотеки: psutil, scikit-learn, joblib.

5) Доступ: мережевий або локальний.

Порядок розгортання:

1) Встановлення:

Необхідно завантажити програмні файли системи та встановити бібліотеки за допомогою pip install -r requirements.txt. Після цього слід перевірити наявність усіх модулів, необхідних для запуску системи.

2) Налаштування параметрів:

Задати параметри роботи системи: інтервал збору метрик та порогові значення. Налаштувати інтерфейс та email-сповіщення.

3) Запуск:

Виконати: python <назва_модуля>.py.

4) Навчання моделі:

Після накопичення достатньої кількості показників запускається модуль python aml.py train. Він використовує алгоритм Isolation Forest для навчання моделі на основі зібраних даних моніторингу. Модель зберігається у файлі.

5) Інтерфейс:

Перейти за посиланням у браузері.

Структура модулів системи:

1) dc.py (Документація) – формування інструкції користувача.

2) cr.py (Безпека) – авторизація користувачів на основі JWT-токена та запис у лог-файл.

3) cn.py (Конфігурація) – налаштування параметрів системи.

4) mon.py (Моніторинг) – збір метрик та запис у лог-файл.

5) ad.py (Аналіз даних) – аналіз зібраних метрик.

6) ar.py (Розподіл ресурсів) – виконання допоміжних дій для зменшення навантаження.

7) lg.py (Логування) – запис подій у лог-файл.

8) aml.py (Виявлення аномалій) – навчання та застосування моделі IF.

9) no.py (Сповіщення) – формування та надсилання повідомлень на пошту.

10) config.json – файл для зберігання інтервалу та порогових значень.

Інструкція користувача:

Спочатку необхідно перейти до модуля dc.py для ознайомлення з порядком дій:

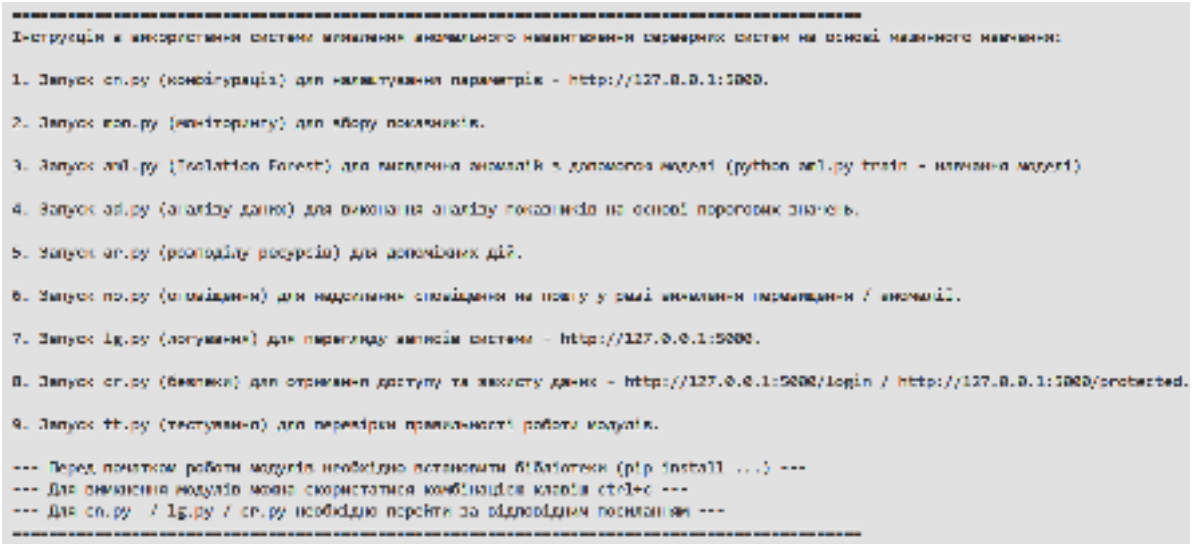

Рисунок 5.1 – Порядок дій

Після запуску cr.py потрібно перейти за відповідними посиланнями, виконати логін для отримання JWT-токену та ввести його для отримання доступу:



Рисунок 5.2 – Логін



Рисунок 5.3 – Отримання токену

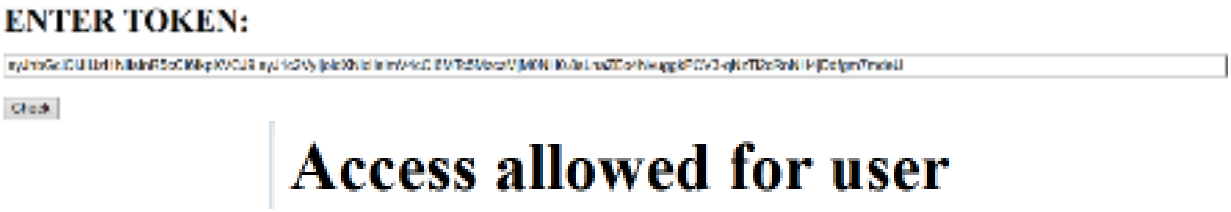

Рисунок 5.4 – Введення токену та отримання доступу

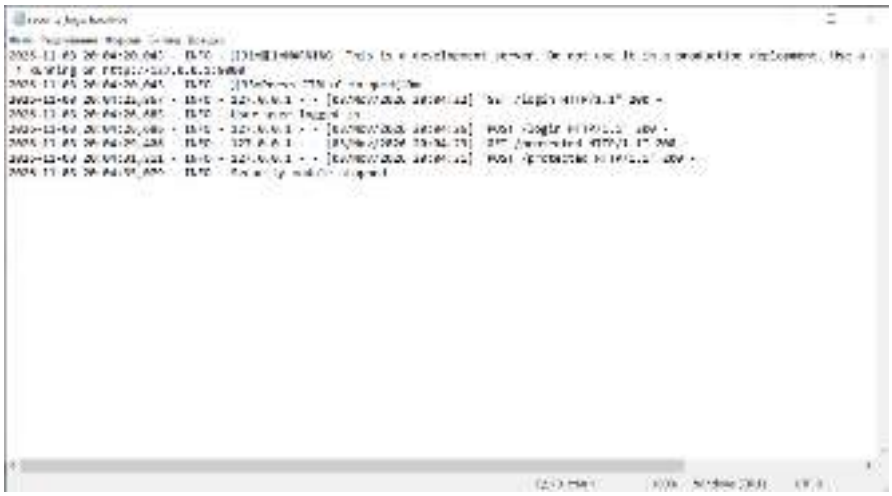

Рисунок 5.5 – Запис інформації

У результаті успішної авторизації система генерує JWT-токен, після введення якого надається доступ. Результати фіксуються у файлі security_logs.log.

Перед запуском всіх наступних модулів потрібно оновити або залишити ту саму конфігурацію для вказаних параметрів за відповідним посиланням:

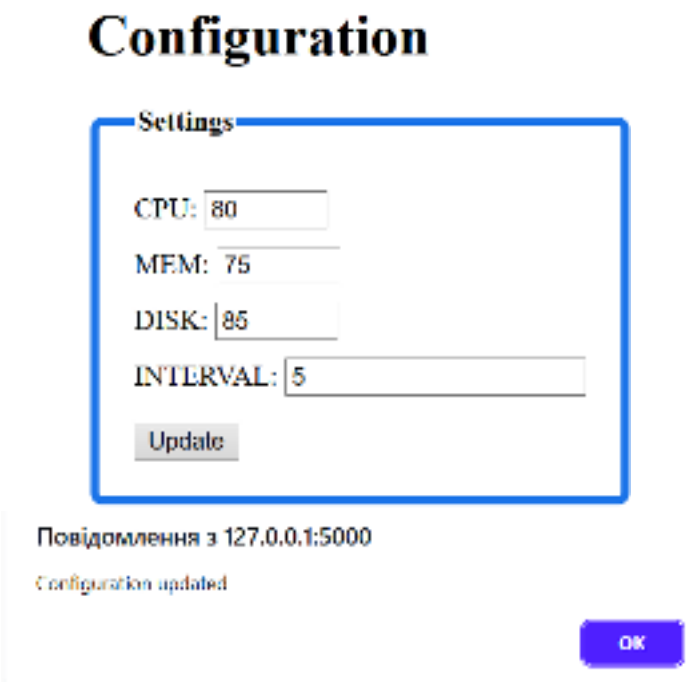

Рисунок 5.6 – Оновлення конфігурації

На цьому етапі задаються основні параметри роботи системи: інтервал та порогові значення. У разі успішного оновлення виводиться відповідне повідомлення та дані записуються у config.json.

Далі потрібно запустити модулі mon.py (збір показників), ad.py (аналіз даних), ar.py (розподіл ресурсів):

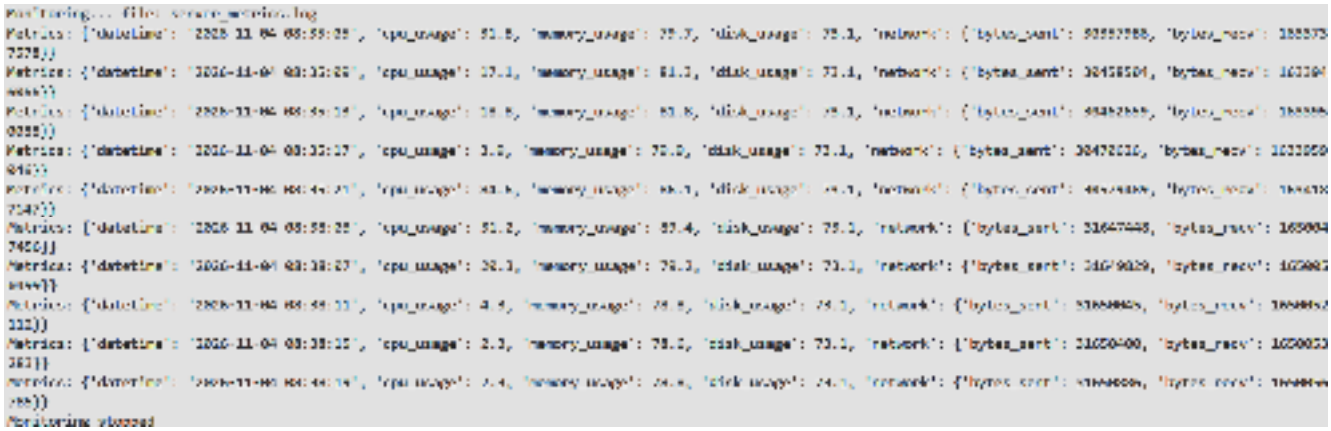

Рисунок 5.7 – Моніторинг



Рисунок 5.8 – Розподіл ресурсів



Рисунок 5.9 – Аналіз

Отримані під час моніторингу дані містять поточні метрики з датою: cpu_usage (використання процесора), memory_usage (використання пам’яті), disk_usage (використання дискового простору), bytes_sent (вихідний трафік), bytes_recv (вхідний трафік). Розподіл ресурсів забезпечується звільненням пам’яті з допомогою зупинення відповідного процесу. У результаті роботи порогового аналізу ad.py формується висновок про стан перевантаження з його описом.

Перегляд логів здійснюється за відповідним посиланням після запуску lg.py:



Рисунок 5.10 – Логування даних



Рисунок 5.11 – Перегляд логів

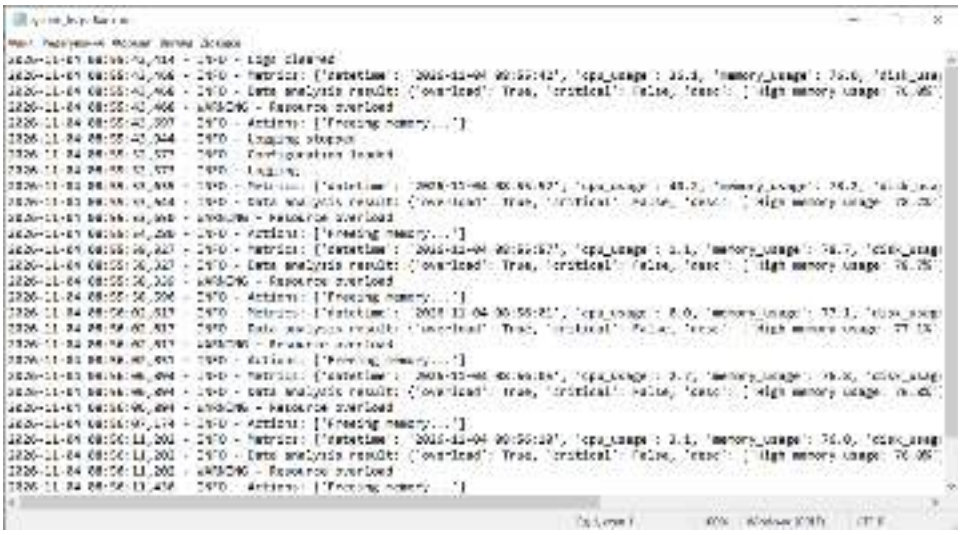

Рисунок 5.12 – Запис логів у файл

У файлах відображено події, які містять історію роботи модулів, часові мітки звернень, підготовлені для подальшого навчання моделі машинного навчання.

Після цього виконується модуль виявлення аномалій aml.py на основі зібраних показників моніторингом. Для навчання та збереження моделі потрібно прописати команду python aml.py train:

```
Model not found. Train first
Model successfully trained and saved
Records: 50
```

Рисунок 5.13 – Навчання та збереження моделі

При додаванні нових показників моніторингом можна просто запустити aml.py у звичайному режимі для їх аналізу та виявлення відхилень:

```
Date: 2026-11-04 09:27:37
CPU: 8.5 %
MEM: 84.9 %
DISK: 73.2 %
Isolation Forest: No anomalous load detected
Threshold method: Overload detected


Date: 2026-11-04 09:27:41
CPU: 68.8 %
MEM: 85.2 %
DISK: 73.2 %
Isolation Forest: Anomalous load detected
Threshold method: Overload detected


Results:
Records: 20
Anomalies detected (Isolation Forest): 10
Overloads detected (Threshold method): 20
Matches: 10
```

Рисунок 5.14 – Виявлення відхилень та порівняння результатів

```
No new records found
```

Рисунок 5.15 – Відсутність записів

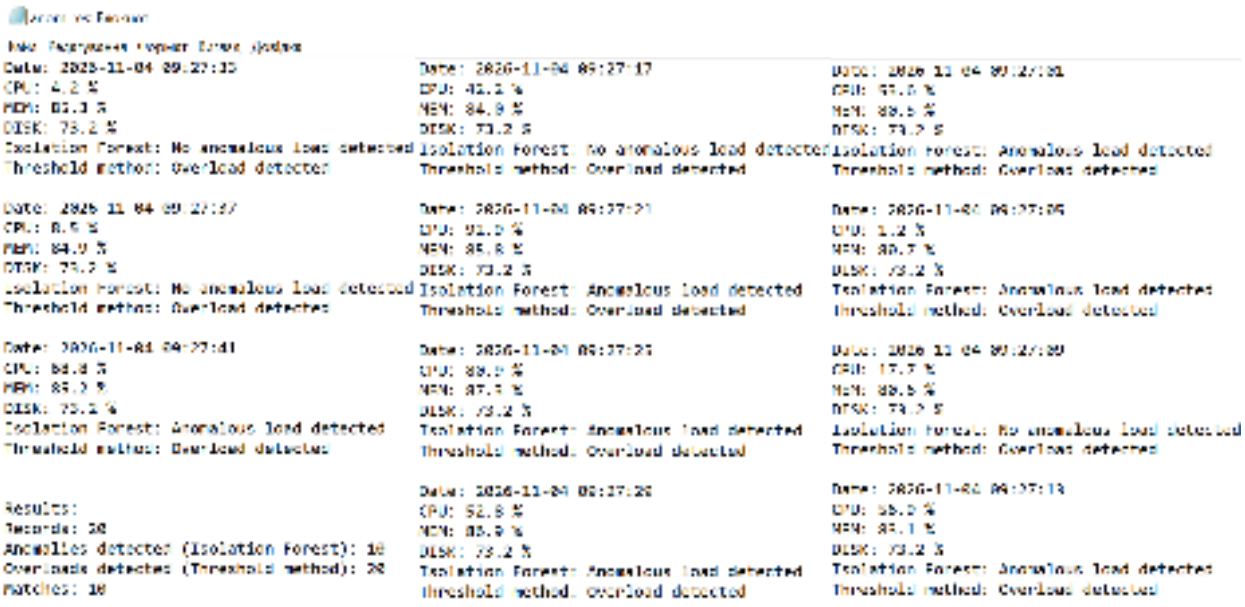

Рисунок 5.16 – Запис інформації у файл

На рисунку 5.14 представлено консольний вивід результатів порівняльного аналізу на вибірці з 20 записів. Отримані дані демонструють, що при стабільно високому використанні пам’яті та низькому навантаженні процесора пороговий метод фіксує перевантаження у всіх 20 випадках, тоді як алгоритм Isolation Forest розпізнає цей стан як нормальний.

При різкому стрибку завантаження центрального процесора до 68,8 % обидва методи виявляють відхилення, що в підсумку дало 10 збігів та дозволило відсіяти 50 % фонових спрацювань. У разі відсутності нових даних у лог-файлі виводиться відповідне повідомлення. Всі виявлені відхилення та підсумкова статистика зберігаються у файлі anom_res.log.

Конфігурацію також можна змінити у файлі config.json для встановлення нових параметрів:



Рисунок 5.17 – Перегляд конфігурації у файлі

У разі зміни цих параметрів у файлі система автоматично застосовує оновлені обмеження.

Вказавши пошту та пароль від неї, можна скористатися модулем сповіщення no.py:



Рисунок 5.18 – Сповіщення



Рисунок 5.19 – Перелік повідомлень



Рисунок 5.20 – Вміст повідомлення

У результаті роботи модуля сповіщень отримується електронний лист, який містить причину, значення та стан перевантаження. Також у консоль виводиться інформація про відправлення повідомлення.

Після виконання всіх модулів виконується тестування:



Рисунок 5.21 – Unit testing

Результати модульного тестування підтверджують успішне проходження всіх перевірок правильності роботи модулів.

Для підтвердження авторства та ідентифікації ПЗ під час запуску системи передбачено виведення інформаційного вікна авторського права, що містить відомості про заклад вищої освіти, кафедру, тему випускної кваліфікаційної роботи, автора та наукового керівника:

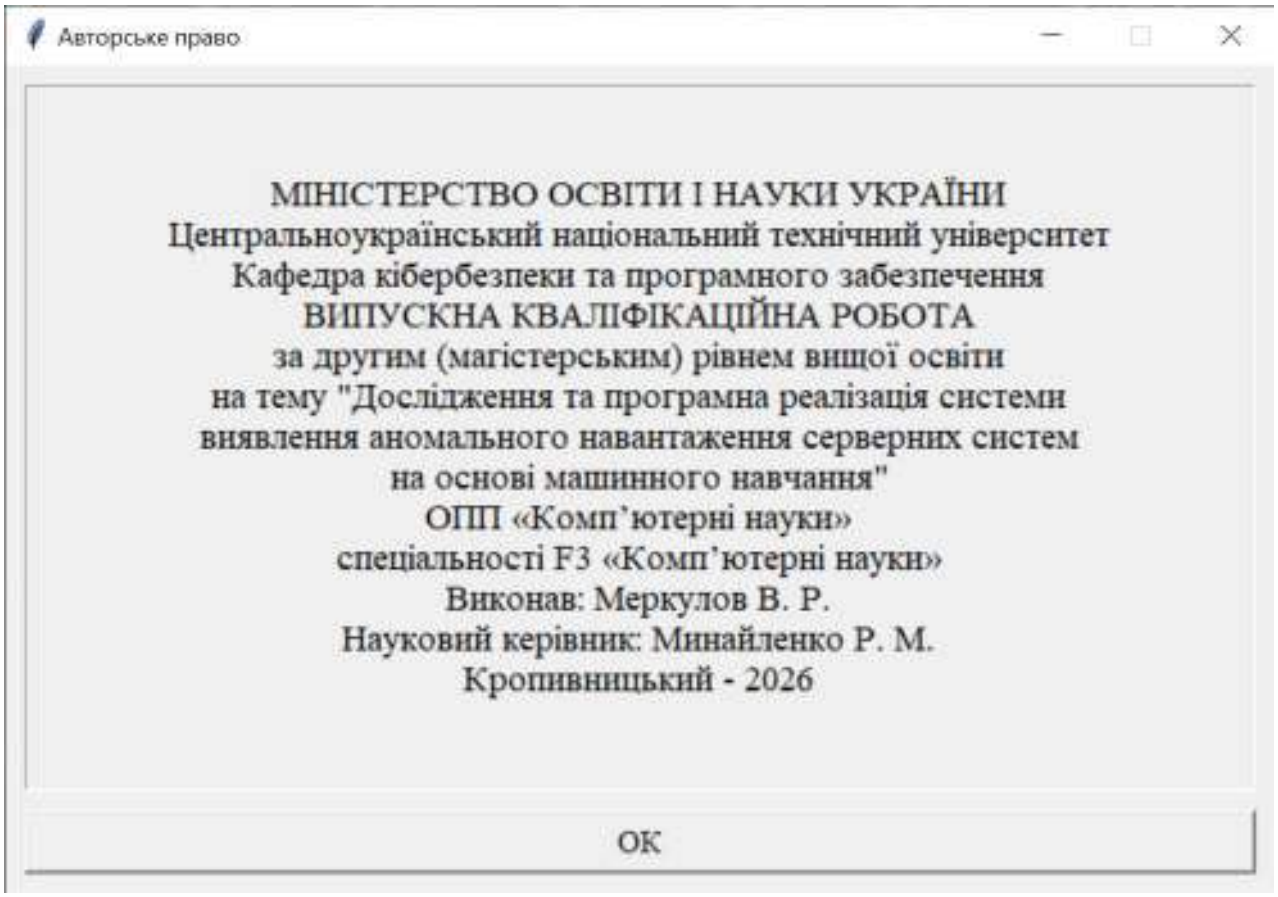

Рисунок 5.22 – Форма авторського права

Кожен модуль виконує задані функції та відіграє важливу роль. Інтерфейс надає можливості для оновлення конфігурації, забезпечення безпеки та перегляду логів. Правильність роботи системи виявлення аномального навантаження серверних систем можна перевірити за змінами показників, порівнянням результатів та логами подій.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи виявлення аномального навантаження серверних систем на основі машинного навчання.

Метою розробки є дослідження та програмна реалізація системи виявлення аномального навантаження серверних систем на основі машинного навчання.

Об'єктом дослідження є процес виявлення аномального навантаження серверних систем на основі машинного навчання.

Предметом дослідження є методи виявлення аномального навантаження серверних систем на основі машинного навчання.

Методами дослідження є системний аналіз для вивчення принципів функціонування систем, порівняльний аналіз для зіставлення результатів, статистичний аналіз для опрацювання показників, методи машинного навчання.

Наукова новизна одержаних результатів полягає у тому, що:

1) Удосконалено підхід до виявлення аномального навантаження серверних систем шляхом використання алгоритму Isolation Forest для аналізу показників використання центрального процесора, оперативної пам'яті та дискового простору, що дозволяє виявляти нетипові зміни у сукупності метрик та визначати приховані аномалії (підрозділи 3.1, 4.1).

2) Удосконалено процес обробки даних моніторингу та процес інформування про аномальне навантаження серверної системи на основі використання результатів аналізу показників для формування повідомлень про виявлені перевантаження та аномалії (принцип обробки наведено у підрозділі 3.1, схему, алгоритм та опис функції chk_an() наведено у підрозділі 4.1).

7 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

7.1 Вступ

Охорона праці є складовою організації діяльності підприємства та спрямована на збереження життя, здоров’я і працездатності працівників у процесі трудової діяльності. Її основними завданнями є створення безпечних умов праці, запобігання нещасним випадкам і професійним захворюванням, а також зменшення впливу небезпечних і шкідливих виробничих факторів. Ці принципи закріплені в базовому законодавстві про охорону праці та деталізуються у нормативах і посібниках. Для фахівців, системних адміністраторів та інженерів ключове значення мають вимоги безпеки під час роботи з екранними пристроями [75, 76, 77].

У цьому розділі розглядаються умови праці у відділі програмістів, де працюють 8 осіб, а площа приміщення становить 60 м². Основними видами діяльності є розроблення, налагодження, тестування програмного забезпечення, опрацювання технічної документації, аналіз даних та робота з комп’ютерною технікою. Праця виконується сидячи й супроводжується тривалим зоровим та розумовим навантаженням.

Метою розділу є аналіз санітарно-гігієнічних умов праці, визначення основних ризиків, розроблення заходів з їх зменшення, розгляд правил електробезпеки та протипожежної профілактики, а також виконання розрахунку захисного занулення електроустановки потужністю 3,8 кВт.

Правовою основою є Закон України «Про охорону праці», який визначає обов’язки роботодавця щодо створення безпечних умов, інформування працівників, навчання та організації профілактичних заходів. Для робочих місць з екранними пристроями застосовуються вимоги, затверджені наказом Мінсоцполітики України № 207. Питання мікроклімату й освітлення розглядаються з урахуванням санітарних та будівельних норм [75, 78, 79, 80].

7.2 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Для аналізу приймається офісне приміщення відділу програмістів площею 60 м², у якому працюють 8 осіб. Кожне робоче місце обладнане персональним комп'ютером, монітором, клавіатурою, мишею та офісним кріслом. Додатково можуть використовуватися шафи для документації, мережеве обладнання та принтер.

Площа на працівника визначається за формулою $S_1 = S/N = 7{,}5$ м^2/особу, де S – площа приміщення, N – кількість працівників.

Отримане значення характеризує планування, але не замінює перевірки ширини проходів, розміщення меблів, доступу до евакуаційних виходів та можливості змінювати робочу позу. Площа 7,5 м²/особу дозволяє забезпечити достатній простір для вільних рухів та зміни робочого положення працівника відповідно до вимог [79].

Робочі столи доцільно розмістити так, щоб світло падало збоку, а монітори не розташовувалися безпосередньо навпроти вікон або яскравих світильників. Проходи між столами та до виходу мають бути вільними. Кабелі живлення та мережеві кабелі потрібно прокладати так, щоб вони не створювали перешкод для пересування.

Під час розроблення програмного забезпечення системи виявлення аномального навантаження серверних систем основним обладнанням є персональний комп'ютер. Робота розробника передбачає тривале використання монітора, клавіатури та інших пристроїв, тому необхідно дотримуватися вимог охорони праці.

Для програмістів характерні тривале перебування в сидячому положенні, статичне навантаження на м'язи спини, шиї та рук, напруження зору, недостатнє або надмірне освітлення, відблиски на екранах, несприятливий мікроклімат, шум від систем охолодження комп'ютерів, а також небезпека ураження електричним струмом.

До психофізіологічних факторів належать інтенсивна розумова робота, необхідність тривало концентрувати увагу, відповідальність за результати програмування та можливість роботи в умовах стислих строків. Для зменшення їх впливу необхідно раціонально планувати завдання, не допускати систематичних понаднормових навантажень та забезпечувати відпочинок.

Освітлення має забезпечувати чітке сприйняття тексту на екрані та паперових документах. У приміщенні доцільно поєднувати природне освітлення через вікна зі штучним освітленням. Світильники слід розташовувати так, щоб їх яскраві елементи не відбивалися в моніторах і не потрапляли в поле зору працівників.

Нормовані показники освітленості визначаються за призначенням приміщення, характером зорової роботи та чинною редакцією ДБН В.2.5-28:2018. Необхідно регулярно очищати світильники, вікна, замінювати несправні лампи та застосовувати локальне освітлення [78].

Мікроклімат робочої зони характеризується температурою повітря, відносною вологістю, швидкістю його руху та температурою навколишніх поверхонь. Для офісної роботи важливо підтримувати комфортні параметри, не допускати протягів і перегрівання. Нормативні параметри визначаються з урахуванням категорії робіт за енерговитратами та періоду року відповідно до ДСН 3.3.6.042-99 для категорії робіт Іа.

Оскільки програмісти переважно виконують роботи сидячи, для забезпечення комфортних умов слід використовувати систему опалення та вентиляції. Провітрювання необхідно проводити регулярно, уникаючи прямого потоку холодного повітря на працівників. Температуру, вологість, швидкість руху повітря слід контролювати відповідними приладами.

Джерелами шуму є вентилятори системних блоків, серверне й мережеве обладнання, принтери та кондиціонери. Для зниження шуму рекомендується використовувати справне обладнання, очищати системи охолодження та розміщувати принтери на відстані від постійних робочих місць. Рівні шуму мають відповідати санітарним вимогам наказу № 207 [79, 80].

7.3 Розробка заходів з поліпшення стану охорони праці

Організаційні заходи спрямовані на формування безпечної поведінки працівників. До початку роботи кожен працівник повинен пройти вступний інструктаж та інструктаж на робочому місці, ознайомитися з правилами експлуатації комп'ютерної техніки, порядком дій у разі аварійної ситуації та правилами пожежної безпеки. Роботодавець має забезпечити навчання, перевірку знань з охорони праці та інформування про небезпечні й шкідливі фактори.

Для відділу з восьми програмістів доцільно призначити відповідальну особу за організацію охорони праці та визначити порядок повідомлення про несправності. Працівники повинні негайно повідомляти керівника про пошкодження кабелів, розеток, корпусів пристроїв, появу запаху горілої ізоляції або нагрів обладнання.

Режим праці необхідно організувати так, щоб працівники мали можливість робити перерви, змінювати вид діяльності та періодично відводити погляд від екрана. Перерви слід планувати протягом робочого дня відповідно до характеру навантаження та вимог нормативних документів [75, 79].

Кожне робоче місце необхідно обладнати столом достатньої площі та стійким кріслом. Ступні працівника мають спиратися на підлогу або підставку, передпліччя мають вільно лежати на поверхні столу, а клавіатура й миша – перебувати в зоні легкої досяжності рук. Монітор потрібно встановлювати перед працівником, щоб уникати постійного повороту голови.

Відстань до екрана та його висоту слід налаштовувати індивідуально. Зображення має бути стабільним, яскравість і контрастність – відрегульованими відповідно до освітлення приміщення. Необхідно уникати відблисків від вікон та світильників.

Для підтримання належного стану приміщення потрібно проводити регулярне вологе прибирання, очищення вентиляційних решіток, видалення пилу з корпусів обладнання. Очищення внутрішніх компонентів комп'ютера виконують після його відключення від електромережі.

Електричні розетки, подовжувачі, мережеві фільтри слід використовувати відповідно до їх навантаження. Не допускається підключати надмірну кількість потужних пристроїв до одного подовжувача, використовувати пошкоджені кабелі або самостійно ремонтувати електрообладнання. Захисні провідники не можна від'єднувати або використовувати як робочий нульовий провідник [79].

7.4 Техніка безпеки та протипожежна профілактика

Електричне обладнання восьми робочих місць програмістів живиться від мережі змінного струму та становить потенційну небезпеку в разі пошкодження ізоляції, несправності розеток або порушення захисного з'єднання корпусів. Ураження електричним струмом може призвести до судом, опіків, порушення дихання та серцевої діяльності. Тому електрообладнання повинно експлуатуватися у справному стані, а його підключення має відповідати вимогам електробезпеки [75, 76, 77].

Перед початком роботи необхідно оглянути кабелі, вилки, розетки та корпуси пристроїв. Забороняється торкатися електрообладнання мокрими руками, працювати з пристроями з пошкодженою ізоляцією, самостійно відкривати блоки живлення або виконувати ремонт під напругою. У разі появи іскріння, запаху горілої ізоляції чи надмірного нагрівання пристрій потрібно безпечно відключити від мережі та повідомити відповідальну особу.

Металеві корпуси обладнання класу I мають бути приєднані до захисного провідника. У сучасних електроустановках застосовуються системи заземлення TN, TT або IT. Не допускається самовільно встановлювати перемички між робочим нульовим та захисним провідниками в розетках або змінювати схему електропроводки.

Під час експлуатації комп'ютерної техніки слід дотримуватися інструкцій виробників, не перекривати вентиляційні отвори системних блоків, моніторів, не розміщувати на обладнанні сторонні предмети. Підключення й відключення периферійних пристроїв потрібно виконувати відповідно до інструкцій.

Після завершення роботи необхідно вимкнути обладнання та перевірити, чи не залишилися ввімкненими пристрої, які не повинні працювати поза робочим часом. Вимоги № 207 передбачають очищення екранних пристроїв від пилу перед початком роботи та їх відключення після завершення роботи [79].

Основними причинами виникнення пожеж у комп'ютерному приміщенні можуть бути коротке замикання, перевантаження електромережі, перегрівання блоків живлення, пошкодження ізоляції кабелів, несправність подовжувачів або порушення правил експлуатації електрообладнання.

Для запобігання пожежам необхідно підтримувати електрообладнання у справному стані, не перевантажувати електричні мережі, не залишати без нагляду обладнання та не зберігати горючі матеріали поблизу джерел тепла. Евакуаційні проходи й виходи мають бути вільними, двері – доступними для відчинення, а плани евакуації та знаки безпеки – помітними.

У приміщенні відділу площею 60 м², де одночасно перебувають 8 працівників, необхідно передбачити первинні засоби пожежогасіння відповідно до категорії приміщення, площі, наявного обладнання та чинних правил пожежної безпеки. Тип, кількість і місце розташування вогнегасників визначаються відповідальною особою з урахуванням нормативних вимог.

У разі виявлення загоряння працівник повинен повідомити інших осіб, викликати службу за номером 101. Не можна гасити електроустановки під напругою водою.

7.5 Розрахункова частина

Необхідно виконати розрахунок захисного занулення електроустановки потужністю 3,8 кВт при однофазному живленні напругою 220 В. Для розрахунку приймається мідний кабель довжиною L = 20 м з перерізом жил S = 4 мм², питомий опір міді $\rho = 0{,}0175$ Ом·мм²/м, коефіцієнт потужності $\cos\varphi = 0{,}95$, умовний опір джерела живлення $Z_{дж} = 0{,}2$ Ом та автоматичний вимикач з номінальним струмом 20 А.

Робочий струм визначається за формулою:

$$I_{\text{р}} = \frac{P}{(U \cdot cos\varphi)} = \frac{3800}{220 \cdot 0{,}95} = 18{,}18\ A \approx 18{,}2\ A$$

Отже, розрахунковий робочий струм становить приблизно 18,2 А. Для прикладу прийнято автоматичний вимикач на 20 А, номінальний струм якого перевищує робочий струм навантаження. Вибір апарата захисту повинен враховувати допустимий струм кабелю, пускові струми та характеристики навантаження.

У разі замикання фазного провідника на металевий корпус електроустановки утворюється коло струму пошкодження. У спрощеному розрахунку опір мідних фазного та захисного провідників визначається за формулою:

$$R_{\text{пров}} = \frac{2 \cdot \rho \cdot L}{S} = \frac{2 \cdot 0{,}0175 \cdot 20}{4} = 0{,}175\ \text{Ом}$$

З урахуванням умовного опору джерела живлення та контактних з’єднань повний опір визначається за формулою:

$$Z_{\text{петлі}} = R_{\text{пров}} + Z_{\text{дж}} = 0{,}175 + 0{,}2 = 0{,}375\ \text{Ом}$$

Отримане значення є оцінкою для прийнятих умов. Реактивний опір, температура провідників та перехідні опори контактів можуть впливати на результат.

Струм однофазного короткого замикання визначається за формулою:

$$I_{\text{к}} = \frac{U}{Z_{\text{петлі}}} = \frac{220}{0{,}375} = 586{,}66\ \text{А} \approx 586{,}7\ \text{А}$$

Для автоматичного вимикача умовний струм миттєвого електромагнітного розчеплення для попередньої оцінки приймається на верхній межі діапазону:

$$I_{\text{а}} = 10 \cdot 20 = 200\ \text{А}$$

Порівняння дає $I_{\text{к}}$ = 586,7 А > $I_{\text{а}}$ = 200 А. Розрахунковий струм пошкодження перевищує умовний струм миттєвого спрацювання автомата.

Остаточне підтвердження потребує перевірки фактичного повного опору петлі, характеристики конкретного автомата та нормативного часу автоматичного відключення для відповідної системи заземлення.

Захисне занулення полягає в електричному приєднанні відкритих частин електроустановки до заземленої нейтральної точки джерела живлення через захисний провідник. У разі пробою ізоляції та появи напруги на корпусі створюється коло струму пошкодження, унаслідок чого апарат захисту повинен автоматично відключити пошкоджену ділянку. У сучасних мережах відповідні функції реалізуються системою захисних провідників PE/PEN [81, 82].

Вигляд електричної схеми занулення:



Рисунок 7.1 – Електрична схема занулення

На схемі показано джерело живлення із заземленою нейтраллю, фазний провідник L, робочий нульовий провідник N, автоматичний вимикач QF, електроустановку, її металевий корпус, захисний провідник PE та повторне заземлення, що забезпечують шлях струму пошкодження до заземленої нейтральної точки.

Висновки

У розділі розглянуто питання охорони праці та техніки безпеки для відділу програмістів, у якому працюють 8 осіб і який займає приміщення площею 60 м². Питома площа становить 7,5 м² на одного працівника. Цей показник є вихідною характеристикою планування, а остаточна оцінка безпечності розміщення робочих місць має враховувати конфігурацію меблів, проходів та евакуаційних шляхів.

Визначено основні небезпечні та шкідливі фактори: тривале сидіння, зорове навантаження, можливі відхилення параметрів мікроклімату, шум обладнання, ризики ураження електричним струмом та виникнення пожежі. Запропоновано організаційні, ергономічні, санітарно-гігієнічні та технічні заходи для зменшення впливу цих факторів.

Розглянуто вимоги безпечної експлуатації комп'ютерної техніки, електрообладнання та протипожежної профілактики. Наголошено на необхідності інструктажів, підтримання справного стану електромережі, забезпечення вільних евакуаційних шляхів та облаштування робочих місць.

Виконано розрахунок захисного занулення електроустановки потужністю 3,8 кВт за параметрами однофазної мережі 220 В. Отримано робочий струм 18,2 А, оціночний опір петлі 0,375 Ом та струм пошкодження 586,7 А. При використанні автоматичного вимикача 20 А умова за струмом спрацювання виконується.

Запропоновані заходи сприяють формуванню безпечних умов праці у відділі програмістів, збереженню здоров'я працівників та зниженню ризику аварійних ситуацій.

8 ОСНОВНІ ВИСНОВКИ

У процесі виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти було використано знання та навички, отримані під час вивчення дисциплін з програмування та машинного навчання. Було досліджено та реалізовано систему виявлення аномального навантаження серверних систем на основі машинного навчання.

Рішення даного завдання полягало у вирішенні наступних задач:

– проведено аналіз існуючих систем моніторингу та навантаження;

– досліджено функціонування алгоритму Isolation Forest;

– реалізовано модуль моніторингу, аналізу та виявлення аномалій;

– впроваджено модуль сповіщення, логування та конфігурацію;

– виконано порівняння результатів та проведено тестування системи.

Особливу увагу приділено модулю машинного навчання, який використовує алгоритм Isolation Forest для визначення аномального навантаження.

Розроблена система дає змогу збирати та аналізувати показники використання ресурсів серверної системи, виявляти аномальні стани та інформувати про виявлені перевантаження.

Програма реалізована на мові високого рівня Python і може використовуватися на ПЕОМ з ОС Windows 10/11.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача.

У підсумку проведеного дослідження отримано практичний результат у вигляді програмної реалізації системи виявлення аномального навантаження серверних систем на основі машинного навчання. Подальші дослідження можуть бути спрямовані на розширення набору показників, оцінювання точності виявлення аномалій та вдосконалення алгоритмів аналізу навантаження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kumar J., Singh A. K., Mohan A., Buyya R. Machine Learning for Cloud Management : monograph. Boca Raton : CRC Press, 2021. 182 p.

2. Banu S., Raghunathan S., Mavaluru D., Mustafa A. S. Anomaly Detection: Techniques and Applications : book. New York : Nova Science Publishers, 2021. 190 p.

3. Меркулов В., Щербина С. The Development of Artificial Intelligence in the United Kingdom. *Наукові записки*. Кропивницький : ЦНТУ, 2024. Вип. 32. С. 45–49.

4. Богаченко С. В., Меркулов В. Р. Дослідження та розробка методів підвищення енергоефективності моделей машинного навчання в умовах обмежених ресурсів // *Інформаційна безпека та комп'ютерні технології* : матеріали IX Міжнар. наук.-практ. конф. Кропивницький : ЦНТУ, 2026. С. 103.

5. Меркулов В. Р. Метод цифрової трансформації механізму моніторингу ефективності спеціальних обмежувальних заходів в системі Міністерства оборони України : наук. робота, подана на 1 етап Всеукр. конкурсу студент. наук. робіт / Центральноукраїнський національний технічний університет. Кропивницький, 2024. 29 с.

6. Vacca J. R. Cybersecurity and IT Infrastructure Protection : monograph. Waltham : Elsevier, 2014. 381 p.

7. Monitoring Tool Market. URL: https://www.futuremarketinsights.com/reports/monitoring-tool-market (дата звернення: 09.10.2026).

8. Machine Learning Anomaly Detection. URL: https://learn.netdata.cloud/docs/netdata-ai/anomaly-detection (дата звернення: 09.10.2026).

9. Netdata Pros and Cons. URL: https://www.g2.com/products/netdata/reviews?qs=pros-and-cons (дата звернення: 09.10.2026).

10. Anomaly detection. URL: https://docs.opensearch.org/latest/observing-your-data/ad/index (дата звернення: 09.10.2026).

11. OpenSearch. URL: https://theartofcto.com/technologies/opensearch (дата звернення: 09.10.2026).

12. OpenSearch review. URL: https://faun.dev/toolbox/opensearch/review (дата звернення: 09.10.2026).

13. Anomaly detection. URL: https://docs.dynatrace.com/docs/dynatrace-intelligence/anomaly-detection (дата звернення: 09.10.2026).

14. Dynatrace pros and cons. URL: https://www.peerspot.com/products/dynatrace-pros-and-cons (дата звернення: 09.10.2026).

15. Dynatrace Review: Pros, Cons, Pricing & Verdict (2026). URL: https://www.siit.io/tools/trending/dynatrace-review (дата звернення: 09.10.2026).

16. Datadog Watchdog. URL: https://docs.datadoghq.com/watchdog (дата звернення: 09.10.2026).

17. Datadog Pros and Cons. URL: https://www.g2.com/products/datadog/reviews (дата звернення: 09.10.2026).

18. Datadog User Reviews. URL: https://www.softwareadvice.com/bi/datadog-profile/reviews (дата звернення: 09.10.2026).

19. Anomaly detection. URL: https://docs.newrelic.com/docs/alerts/create-alert/set-thresholds/anomaly-detection (дата звернення: 09.10.2026).

20. New Relic Pros and Cons. URL: https://www.g2.com/products/new-relic/reviews?qs=pros-and-cons (дата звернення: 09.10.2026).

21. New Relic User Reviews. URL: https://www.softwareadvice.com/devops/new-relic-one-profile/reviews (дата звернення: 09.10.2026).

22. Grafana Machine Learning. URL: https://grafana.com/docs/grafana-cloud/ai-tools/machine-learning/#grafana-machine-learning (дата звернення: 09.10.2026).

23. Grafana Review: Pros, Cons, Features, and Pricing. URL: https://thectoclub.com/tools/grafana-review (дата звернення: 09.10.2026).

24. Detect metric anomalies. URL: https://www.elastic.co/docs/solutions/observability/infra-and-hosts/detect-metric-anomalies (дата звернення: 09.10.2026).

25. Elastic Observability Pros and Cons. URL: https://www.g2.com/products/elastic-observability/reviews?qs=pros-and-cons (дата звернення: 09.10.2026).

26. Splunk IT Service Intelligence. URL: https://help.splunk.com/en/splunk-it-service-intelligence (дата звернення: 09.10.2026).

27. Splunk ITSI (IT Service Intelligence) pros and cons. URL: https://www.peerspot.com/products/splunk-itsi-it-service-intelligence-pros-and-cons (дата звернення: 09.10.2026).

28. Using CloudWatch anomaly detection. URL: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch_Anomaly_Detection.html (дата звернення: 09.10.2026).

29. 3 Pros and Cons of Amazon CloudWatch. URL: https://www.chaossearch.io/blog/pros-and-cons-amazon-cloudwatch (дата звернення: 09.10.2026).

30. AI and Machine Learning with Sumo Logic. URL: https://www.sumologic.com/help/docs/get-started/ai-machine-learning (дата звернення: 09.10.2026).

31. Sumo Logic Pros and Cons. URL: https://www.g2.com/products/sumo-logic/reviews?qs=pros-and-cons (дата звернення: 10.10.2026).

32. IBM Instana Observability documentation. URL: https://www.ibm.com/docs/en/instana-observability (дата звернення: 10.10.2026).

33. IBM Instana Observability pros and cons. URL: https://www.peerspot.com/products/ibm-instana-observability-pros-and-cons (дата звернення: 10.10.2026).

34. Cisco Observability Platform Anomaly Detection Configuration API. URL: https://developer.cisco.com/docs/cisco-observability-platform/anomaly-detection (дата звернення: 10.10.2026).

35. Cisco Crosswork Network Insights vs Groundcover Observability Platform comparison. URL: https://www.peerspot.com/products/comparisons/cisco-crosswork-network-insights_vs_groundcover-observability-platform (дата звернення: 10.10.2026).

36. Cisco Observability Platform. URL: https://www.gartner.com/reviews/product/cisco-observability-platform (дата звернення: 10.10.2026).

37. Documentation for SolarWinds. URL: https://documentation.solarwinds.com (дата звернення: 10.10.2026).

38. SolarWinds Pros and Cons. URL: https://www.g2.com/products/solarwinds-worldwide-llc-solarwinds-observability/reviews (дата звернення: 10.10.2026).

39. LogicMonitor. URL: https://www.logicmonitor.com (дата звернення: 10.10.2026).

40. LogicMonitor Pros and Cons. URL: https://www.g2.com/products/logicmonitor/reviews (дата звернення: 10.10.2026).

41. ScienceLogic AI Platform. URL: https://sciencelogic.com/platform/overview (дата звернення: 10.10.2026).

42. ScienceLogic AI Platform Pros and Cons. URL: https://www.g2.com/products/sciencelogic-ai-platform/reviews?qs=pros-and-cons (дата звернення: 10.10.2026).

43. BigPanda Docs. URL: https://docs.bigpanda.io (дата звернення: 10.10.2026).

44. BigPanda pros and cons. URL: https://www.peerspot.com/products/bigpanda-pros-and-cons (дата звернення: 10.10.2026).

45. Dell AIOps. URL: https://www.dell.com/en-us/shop/dell-aiops/sl/aiops?hve=Explore+Dell+AIOps (дата звернення: 11.10.2026).

46. Moogsoft Pros and Cons. URL: https://www.g2.com/products/moogsoft/reviews?qs=pros-and-cons (дата звернення: 11.10.2026).

47. IT infrastructure & Application Monitoring Tool - ManageEngine Applications Manager. URL: https://www.manageengine.com/products/applications_manager (дата звернення: 11.10.2026).

48. ManageEngine Applications Manager Pros and Cons. URL: https://www.g2.com/products/manageengine-applications-manager/reviews?qs=pros-and-cons (дата звернення: 11.10.2026).

49. 9 найкращих інструментів хмарного моніторингу (2026). URL: https://www.guru99.com/uk/best-cloud-monitoring-tools.html (дата звернення: 11.10.2026).

50. Site24x7 Pros and Cons. URL: https://www.g2.com/products/site24x7/reviews?qs=pros-and-cons (дата звернення: 11.10.2026).

51. Server monitoring. URL: https://sematext.com/capabilities/server-monitoring (дата звернення: 15.10.2026).

52. 15+ Best Cloud Monitoring Tools of 2020: Pros & Cons Comparison. URL: https://dev.to/sematext/15-best-cloud-monitoring-tools-of-2020-pros-cons-comparison-35k (дата звернення: 15.10.2026).

53. Honeycomb: AI-Ready Observability Platform. URL: https://www.honeycomb.io (дата звернення: 15.10.2026).

54. Honeycomb Pros and Cons. URL: https://www.g2.com/products/honeycomb/reviews?qs=pros-and-cons (дата звернення: 15.10.2026).

55. Anomaly Detection. URL: https://docs.victoriametrics.com/anomaly-detection (дата звернення: 15.10.2026).

56. VictoriaMetrics Pros and Cons. URL: https://www.g2.com/products/victoriametrics/reviews (дата звернення: 18.10.2026).

57. EventSentry Overview. URL: https://www.eventsentry.com/documentation/overview/html/overview.htm (дата звернення: 18.10.2026).

58. EventSentry Pros and Cons. URL: https://www.g2.com/products/eventsentry/reviews?qs=pros-and-cons (дата звернення: 18.10.2026).

59. What is Pandora FMS. URL: https://pandorafms.com/guides/public/books/what-is-pandora-fms (дата звернення: 18.10.2026).

60. Pandora FMS Pros and Cons. URL: https://www.g2.com/products/pandora-fms/reviews?qs=pros-and-cons (дата звернення: 18.10.2026).

61. What is SigNoz. URL: https://signoz.io/docs/what-is-signoz (дата звернення: 18.10.2026).

62. CABOT SOFTWARE & APP. URL: https://www.yachtica.com/en/prodotto/cabot-software-app (дата звернення: 18.10.2026).

63. Cabot vs Nagios. URL: https://stackshare.io/stackups/cabot-vs-nagios (дата звернення: 18.10.2026).

64. Opsview pros and cons. URL: https://www.peerspot.com/products/opsview-pros-and-cons (дата звернення: 18.10.2026).

65. Мова програмування Пайтон. URL: https://www.scribd.com/presentation/741930885/Мова-програмування-Пайтон (дата звернення: 21.10.2026).

66. Python. URL: https://uk.wikipedia.org/wiki/Python (дата звернення: 21.10.2026).

67. The Python standard library. URL: https://docs.python.org/uk/3/library/index.html (дата звернення: 23.10.2026).

68. Isolation Forest. URL: https://en.wikipedia.org/wiki/Isolation_forest (дата звернення: 23.10.2026).

69. Python Features. URL: https://www.geeksforgeeks.org/python/python-features (дата звернення: 23.10.2026).

70. Структурна схема. URL: https://uk.wikipedia.org/wiki/Структурна_схема (дата звернення: 25.10.2026).

71. Structural Diagrams. URL: https://www.geeksforgeeks.org/structural-diagrams-unified-modeling-languageuml (дата звернення: 25.10.2026).

72. Функціональна схема. URL: https://uk.wikipedia.org/wiki/Функціональна_схема (дата звернення: 25.10.2026).

73. What is a Process Flow Diagram. URL: https://lucid.co/diagram/process-map/process-flow-tutorial (дата звернення: 25.10.2026).

74. Flowchart. URL: https://en.wikipedia.org/wiki/Flowchart (дата звернення: 27.10.2026).

75. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. URL: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення: 11.10.2026).

76. Зеркалов Д. В. Охорона праці в галузі : Загальні вимоги : навч. посіб. Київ : Основа, 2011. 551 с.

77. Оришака О. В., Горбачова Г. П., Марченко К. М. Основи охорони праці : навч. посіб. Кропивницький : ЦНТУ, 2022. 175 с.

78. Державні будівельні норми України: ДБН В.2.5-28:2018. Природне і штучне освітлення. URL: https://goo.su/9AkQ (дата звернення: 11.10.2026).

79. Наказ Міністерства соціальної політики України від 14.02.2018 р. № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». URL: https://zakon.rada.gov.ua/laws/show/z0508-18 (дата звернення: 11.10.2026).

80. Постанова Головного державного санітарного лікаря України від 01.12.1999 р. № 42 «Санітарні норми мікроклімату виробничих приміщень» ДСН 3.3.6.042-99. URL: https://zakon.rada.gov.ua/rada/show/va042282-99 (дата звернення: 12.10.2026).

81. Охорона праці. Ч. 1. Захисне заземлення : метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / укл.: О. В. Оришака, Є. К. Солових, В. О. Оришака. Кіровоград : КІСМ, 1997. 20 с.

82. Охорона праці. Ч. 2. Занулення : метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / укл.: О. В. Оришака, Є. К. Солових, В. О. Оришака, А. Є. Солових, С. Е. Катеринич. Кропивницький : ЦНТУ, 2019. 27 с.