Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ**:**
В.о. завідувача кафедри
кібербезпеки та захисту
інформації
_____ Іван ПАРХОМЕНКО
«__» _____ 2025 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань *12 Інформаційні технології*
(шифр і назва галузі знань)

спеціальність *125 Кібербезпека та захист інформації*
(код і назва спеціальності)

освітній ступень *магістр*

освітньо-наукова програма *Кібербезпека*

на тему: «Експертна система вибору антивірусного програмного забезпечення для об’єктів критичної інфраструктури»

Виконавець: студент ІІ курсу, групи КБм-22

_____ **Максим МАЄВСЬКИЙ**
(підпис) (ім’я, прізвище)

	Ім’я, прізвище	**Підпис**
Керівник	Сергій БУЧИК	
Нормоконтроль	Сергій ДАКОВ	

Київ 2025

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«___» _____ 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *125 Кібербезпека і захист інформації*
(код і назва спеціальності)

освітній ступень *магістр*

Здобувача **КБм-22** **Маєвського Максима Олександровича**
(група) (прізвище ім'я по-батькові)

Тема кваліфікаційної роботи Експертна система вибору антивірусного програмного забезпечення для об'єктів критичної інфраструктури

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №4 від 24.10.2024 р.

2. МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень	процес вибору антивірусного програмного забезпечення для захисту інформаційних систем об'єктів критичної інфраструктури.
Предмет досліджень	методи та критерії оцінки ефективності антивірусного програмного забезпечення.
Мета	створення експертної системи для автоматизації вибору оптимального антивірусного програмного забезпечення для об'єктів критичної інфраструктури з використанням багатокритеріального аналізу та експертних оцінок.
Вихідні дані для проведення роботи	методики оцінювання ефективності антивірусного програмного забезпечення.

3. ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна удосконаленні комплексної моделі оцінки антивірусного програмного забезпечення.

Практична цінність автоматизація та оптимізація процесу вибору антивірусного програмного забезпечення, що значно підвищує ефективність захисту об’єктів критичної інфраструктури від кіберзагроз.

4. ЕТАПИ ВИКОНАННЯ РОБОТИ

Найменування етапів робіт	**Строки виконання робіт (початок-кінець)**
Уточнення постановки задачі	25.10.2024 – 20.01.2025
Аналіз літератури та існуючих підходів до захисту критичної інфраструктури	21.01.2025 – 03.02.2025
Дослідження загроз інформаційній безпеці об’єктів критичної інфраструктури	04.02.2025 – 20.02.2025
Огляд сучасних антивірусних рішень та їх функціональних можливостей	21.02.2025 – 03.03.2025
Формування критеріїв оцінки антивірусного програмного забезпечення	04.03.2025 – 13.03.2025
Розробка методології порівняльного аналізу антивірусних рішень	14.03.2025 – 29.03.2025
Розробка математичної моделі процесу прийняття рішень	30.04.2025 – 14.04.2025
Розробка алгоритму роботи експертної системи	15.04.2025 – 22.04.2025
Програмна реалізація експертної системи	23.04.2025 – 09.05.2025
Тестування та валідація експертної системи	09.05.2025 – 12.05.2025
Оформлення пояснювальної записки згідно методичних рекомендацій	13.05.2025 – 18.05.2025
Подача пакету документів на розгляд ЕК	19.05.2025

Завдання видав _____ Сергій БУЧИК
(підпис) (ім’я, прізвище)

Завдання прийняв до виконання _____ Максим МАЄВСЬКИЙ
(підпис) (ім’я, прізвище)

Дата видачі завдання: 25.10.2024 р.

Термін подання кваліфікаційної роботи до ЕК 19.05.2025 р.

УДК 004.027 : 004.056.5

РЕФЕРАТ

Пояснювальна записка до магістерської роботи «Експертна система вибору антивірусного програмного забезпечення для об’єктів критичної інфраструктури»:79 сторінок, 19 рисунків, 35 таблиць та 31 літературне джерело.

Об’єкт дослідження – процес вибору антивірусного програмного забезпечення для захисту інформаційних систем об’єктів критичної інфраструктури від кіберзагроз.

Мета роботи – підвищення ефективності процесу вибору оптимального антивірусного програмного забезпечення для об’єктів критичної інфраструктури шляхом розробки експертної системи, що автоматизує цей процес на основі багатокритеріального аналізу та експертних оцінок.

Методи дослідження базуються на використанні системного аналізу при дослідженні процесів захисту інформації, теорії прийняття рішень при розробці алгоритмів вибору антивірусного програмного забезпечення, методів експертної оцінки при формуванні бази знань та математичного моделювання в створення моделі оцінки ефективності антивірусних рішень.

У роботі досягнуто наступних наукових результатів: розроблено комплексну математичну модель оцінки антивірусного програмного забезпечення. Удосконалено метод багатокритеріального аналізу для вибору антивірусних рішень шляхом введення вагових коефіцієнтів для критеріїв.

Подальші дослідження можуть бути проведені для вдосконалення алгоритмів прийняття рішень з урахуванням нових типів кіберзагроз та адаптації системи для використання в інших сферах інформаційної безпеки.

Ключові слова: експертна система, антивірусне програмне забезпечення, критична інфраструктура, кібербезпека, багатокритеріальний аналіз, прийняття рішень, база знань, інформаційна безпека, оцінка продуктивності, нечітка логіка, математичне моделювання, системний аналіз.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

APT	–	Сучасні постійні загрози
CR	–	Consistency Ratio
CRUD	–	Create, Read, Update, Delete
CVSS	–	Common Vulnerability Scoring System
DDoS	–	Distributed Denial of Service
ДССЗЗІ	–	Державна служба спеціального зв’язку та захисту інформації України
ДСТУ	–	Державний стандарт України
EDR	–	Endpoint Detection and Response
GUI	–	Graphical User Interface
ІТ	–	інформаційні технології
ІУ	–	індекс узгодженості
NGFW	–	Next-Generation Firewall
SIEM	–	Security Information and Event Management
СУБД	–	Система управління базами даних

ВСТУП

Стрімкий розвиток інформаційних технологій та зростання кількості кібератак на об’єкти критичної інфраструктури створюють нові виклики у сфері кібербезпеки. Згідно зі звітом CERT-UA, за останні роки в Україні спостерігається значне зростання частоти та складності таких атак, що підкреслює нагальну потребу в ефективних захисних заходах.

Як вітчизняні, так і міжнародні дослідники активно досліджують питання захисту критичної інфраструктури від кіберзагроз. Вітчизняні вчені Бурячок О.Г., Бурячок В.Б., Дудикевич В.Б. зробили значний внесок у 1990-1991 роках, створивши теоретичні основи систем захисту об’єктів критичної інфраструктури. Крім того, Ланде Д.В., Хорошко В.О. та Архипов О.В. зосередилися на розробці експертних систем та систем підтримки прийняття рішень.

Актуальність теми: необхідність вирішення протиріччя між зростанням складності вибору оптимального антивірусного програмного забезпечення для об’єктів критичної інфраструктури та обмеженими можливостями існуючих підходів до прийняття таких рішень. Існуючі методології не в повній мірі враховують специфіку різних типів об’єктів критичної інфраструктури чи сучасні тенденції розвитку кіберзагроз.

Метою магістерської роботи є створення експертної системи для автоматизації вибору оптимального антивірусного програмного забезпечення для об’єктів критичної інфраструктури з використанням багатокритеріального аналізу та експертних оцінок.

Завдання магістерської роботи:

1. Проаналізувати сучасні антивірусні рішення та їхні можливості.
2. Визначити критерії оцінки антивірусного програмного забезпечення.
3. Розробити математичну модель та алгоритми прийняття рішень.
4. Створити базу знань експертної системи.

5. Реалізувати програмне забезпечення та протестувати його.

Об’єкт дослідження: процес вибору антивірусного програмного забезпечення для захисту інформаційних систем об’єктів критичної інфраструктури.

Предмет дослідження: методи та критерії оцінки ефективності антивірусного програмного забезпечення.

У роботі використовуються методи системного аналізу, теорії прийняття рішень, експертного оцінювання, математичного моделювання та об’єктно-орієнтованого програмування.

Наукова новизна роботи полягає в удосконаленні комплексної моделі оцінки антивірусного програмного забезпечення. Ця модель враховує специфічні потреби об’єктів критичної інфраструктури та вдосконалює метод багатокритеріального аналізу. На відміну від існуючих методів, які в першу чергу зосереджуються на основних функціях продукту, запропонований підхід включає ширший спектр критеріїв, які стосуються таких важливих елементів, як безпека, адаптивність до сучасних кіберзагроз та ефективність виявлення потенційних вразливостей. Крім того, методологія використовує виключно антивірусні рішення зі списку ДССЗЗІ, що підвищує точність оцінки, забезпечує відповідність нормативним вимогам та оптимізує вибір антивірусних рішень для об’єктів критичної інфраструктури.

Практичне значення отриманих результатів полягає в тому, що створена експертна система може оптимізувати вибір антивірусного програмного забезпечення. Така оптимізація дозволить знизити витрати на кібербезпеку та підвищити рівень захисту об’єктів критичної інфраструктури.

Апробація результатів роботи:

Бучик С.С., Маєвський М.О. Система вибору антивірусного програмного забезпечення для об’єктів критичної інфраструктури. VIII Міжнародна науково-практична конференція «Проблеми кібербезпеки інформаційно-комунікаційних систем (PCSICS)».

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ЗАХИСТУ ОБ'ЄКТІВ КРИТИЧНОЇ ІНФРАСТРУКТУРИ

1.1 Поняття та класифікація об'єктів критичної інфраструктури

Об'єкти критичної інфраструктури мають важливе значення для сучасної національної безпеки, економічної стабільності та суспільного добробуту [1]. Стрімкий розвиток інформаційних технологій зробив кібербезпеку ключовим елементом захисту об'єктів критичної інфраструктури. Коли традиційні інфраструктурні системи, такі як електростанції, транспортні мережі та об'єкти водопостачання, виходять з ладу, кібератаки в поєднанні зі шкідливим програмним забезпеченням та несанкціонованим витоком даних можуть призвести до значних збоїв у національних операціях.

У сучасному контексті термін «об'єкти критичної інфраструктури» розширився і включає в себе не лише фізичні споруди та інформаційні системи, а й мережеву інфраструктуру та платформи управління, які є важливими для безперебійного функціонування економіки. Відповідно до Закону України «Про критичну інфраструктуру», ці об'єкти мають вирішальне значення для функціонування державних установ та надання основних послуг, а їх порушення ускладнює або навіть унеможливлює подальший економічний розвиток [2]. Зі збільшенням цифрових загроз зростає значення інформаційної безпеки, навіть незначні порушення в цифровій безпеці можуть спричинити широкомасштабний вплив на традиційні інфраструктурні системи [3].

Класифікація об'єктів критичної інфраструктури була проведена з використанням методології, яка передбачає структурну оцінку, аналіз функціональних характеристик та розуміння соціальних наслідків операційних збоїв [4]. Ідентифікація мереж і технологій інформаційних систем сьогодні має вирішальне значення для кібербезпеки, оскільки вони дозволяють коректно

функціонувати системі управління об’єктами. Методологія оцінки сучасного аналізу ризиків використовує комплексний підхід, який інтегрує економічні фактори з правовими міркуваннями, соціологічними аспектами та технологічними елементами, що включає оцінку загроз і вразливостей безпеки інформаційних систем. Цифровізація систем управління енергетичними об’єктами підвищила вразливість традиційно критично важливої енергетичної інфраструктури, оскільки тепер виробництво та розподіл електроенергії контролюється за допомогою онлайн-технологій. Збій у роботі цих систем може призвести до проблем з електропостачанням і водночас став би відправною точкою для поширення кіберзагроз на всю національну інфраструктуру.

Історично об’єкти критичної інфраструктури класифікувалися на основі структурних характеристик, таких як місце розташування, форма власності та організаційна структура. Сучасні підходи до класифікації зосереджені на функціональних ознаках, що охоплюють різноманітність послуг, які пропонує кожен об’єкт, та його інформаційні аспекти. Метод функціонального аналізу виявляє взаємозалежності між фізичними та кібернетичними елементами, які лежать в основі основних операцій, таких як енергопостачання, транспорт, водопостачання та охорона здоров’я [5]. Відповідно до критеріального підходу, об’єкти сортуються за рівнем стійкості, який поділяється на чотири категорії: національний, регіональний, місцевий та локальний. Для кожного сектору визначаються індивідуальні заходи безпеки для захисту фізичних активів і конфіденційної інформації, що дозволяє зосередити зусилля на системах, які піддаються найбільшому ризику [6; 7].

Кібербезпека має першорядне значення, оскільки сучасні інформаційні системи діють як інструменти управління і стають об’єктами атак. Як кіберзлочинці, так і державні суб’єкти вдаються до кібератак заради політичної чи економічної вигоди. Злиття систем управління критичною інфраструктурою з інформаційними технологіями створює нові ризики, дозволяючи злочинцям отримувати доступ до конфіденційних даних і маніпулювати процесами, потенційно виводячи з ладу важливі компоненти системи. Тому передові

технології кіберзахисту, адаптивні стратегії захисту і регулярні аудити інформаційної безпеки є життєво важливими для управління системами критичної інфраструктури.

Розширена класифікація об’єктів критичної інфраструктури, що включає елементи кібербезпеки, має значну практичну цінність, оскільки дозволяє створювати комплексні стратегії безпеки. Маючи чітко визначені категорії, держава може розробляти ефективні плани модернізації інформаційних систем, створювати оперативні програми для резервних каналів зв’язку та впроваджувати технології виявлення вторгнень. У надзвичайних ситуаціях система швидкого реагування повинна ефективно управляти фізичними компонентами інциденту та потенційними кіберзагрозами, які можуть виникнути внаслідок атак або порушень в інформаційній системі. Така стратегія мінімізує ймовірність широкомасштабних збоїв у роботі системи, пом’якшуючи при цьому негативні економічні та соціальні наслідки [4].

Активне впровадження країнами сучасних стратегій кіберзахисту призводить до створення надзвичайно стійкої критичної інфраструктури. Стандарти Європейського Союзу, а також Сполучених Штатів Америки та інших передових країн ілюструють необхідність поєднання традиційних заходів безпеки з сучасними цифровими технологіями [1]. Така адаптація є особливо актуальною для України, оскільки дозволяє забезпечити достатній захист критичної інфраструктури та цілісність інформаційних систем, які слугують основою для нагляду за всіма критично важливими державними секторами.

Комплексний підхід до захисту критичної інфраструктури, що включає заходи з кібербезпеки, дозволяє державі ефективно протистояти сучасним загрозам. Використання систем моніторингу, проведення оцінки ризиків та розробка стратегій реагування на фізичні та кіберзагрози забезпечують надійний захист. Співпраця між державними установами, операторами критично важливих систем та партнерами з приватного сектору створює єдину мережу захисту, що дозволяє виявляти загрози на ранній стадії та швидко реагувати на них. Така співпраця гарантує, що навіть у разі кібератаки її наслідки можуть бути

швидко локалізовані та усунені, таким чином мінімізуючи шкоду для суспільства та економіки.

Сучасні визначення компонентів критичної інфраструктури мають вирішальне значення для впровадження заходів з ІТ та кібербезпеки. Комплексна система категоризації, що охоплює фізичні та цифрові елементи, дозволяє ефективно організувати об’єкти відповідно до їхньої важливості та потенційної схильності до кіберзагроз. Ця методологія сприяє розробці надійних стратегій управління ризиками, оптимізації ресурсів та впровадженню новітніх технологій інформаційної безпеки, підвищуючи стійкість держави до внутрішніх і зовнішніх загроз. У світлі цифрової трансформації повсякденного життя, створення ґрунтовної системи кібербезпеки є життєво важливим для захисту національної безпеки за допомогою чітко визначених стратегій [8].

Системний аналіз інтегрує економічні, правові, соціологічні та технічні підходи, що дозволяє оцінити окремі компоненти фізичних об’єктів та їхніх інформаційних систем. Сучасні системи кібербезпеки включають механізми раннього виявлення кібератак, резервне копіювання даних, шифрування та постійний моніторинг, які є важливими для усунення значних вразливостей системи [4]. Такий підхід не лише зменшує ризик надзвичайних ситуацій, але й сприяє швидкому реагуванню на загрози, що виникають під час функціонування об’єктів критичної інфраструктури.

Включення кібербезпеки в управління критичною інфраструктурою є життєво важливим для сучасної державної політики. Чітке визначення критеріїв і категорій, що враховують цифровий аспект, допомагає оптимізувати використання ресурсів, сприяє адаптивним стратегіям захисту та забезпечує стійкість державних систем до зростаючих зовнішніх і внутрішніх загроз. Єдиний підхід до вирішення питань фізичної та кібербезпеки покращує умови сталого зростання країни, забезпечуючи економічну стабільність та соціальну безпеку.

1.2 Аналіз загроз інформаційній безпеці об’єктів критичної інфраструктури

Оцінка загроз безпеці критичної інфраструктури потребує більшої уваги, оскільки розвиток цифрових технологій та зростання кількості кібератак завдають значної шкоди національній безпеці та економічній стабільності [9]. Сьогодні критична інфраструктура охоплює не лише фізичні активи, а й інформаційні системи, мережеві платформи та інструменти управління, необхідні для функціонування важливих державних секторів [2]. Низка кіберзагроз, включно зі шкідливим програмним забезпеченням та витоками даних, ставить під загрозу ефективну роботу цих систем, потенційно викликаючи ефект доміно, який впливає як на окремі елементи, так і на всю інфраструктуру в цілому.

Основними загрозами інформаційній безпеці є DDoS-атаки, фішингові кампанії, шкідливе програмне забезпечення та сучасні постійні загрози (APTS). APTS демонструють високий ступінь організованості, здійснюючи тривалі приховані вторгнення в системи. Це дозволяє зловмисникам поступово отримувати контроль над життєво важливими інформаційними ресурсами, які вони можуть використовувати для майбутніх атак або для повного виведення з ладу систем управління [10].

Окрім зовнішніх загроз, важливими є внутрішні ризики. Витоки даних і несанкціоновані зміни в системних процесах можуть виникати через помилки операторів, недбалість працівників або навмисне зловживання. У сучасних умовах, коли системи управління та автоматизації підключені до інформаційних мереж, навіть незначні помилки в конфігурації мережевих протоколів або недбале оновлення програмного забезпечення можуть призвести до серйозних перебоїв в роботі важливих об’єктів.

Розвиток хмарних обчислень та технологій Інтернету речей відкриває нові можливості для кібератак. Одночасні атаки на хмарні сервіси створюють серйозні виклики для швидкого виявлення загроз і впровадження негайного

захисту. Пристроям Інтернету речей часто бракує необхідних ресурсів для розгортання передових заходів безпеки, що робить їх вразливими до атак, які експлуатують апаратні та програмні компоненти.

Сучасне моделювання ризиків – це метод, який використовується для оцінки можливих негативних наслідків кібератак на об’єкти критичної інфраструктури. Воно дозволяє визначити рівні вразливості кожної підсистеми та сприяє створенню адаптивних стратегій захисту [11]. Результатом такого аналізу є виконання ретельних заходів безпеки, включаючи превентивні дії, системи раннього виявлення кібератак, резервне копіювання даних, захист інформації та постійний моніторинг інформаційних ресурсів [4].

Аналіз як зовнішніх, так і внутрішніх загроз є особливо важливим у періоди геополітичної напруженості, коли країна стикається з підвищеною вразливістю від різних викликів [12]. У всеохоплюючій стратегії національної безпеки зазначено, що ці обставини суттєво впливають на кібербезпеку критично важливих об’єктів [13]. Уповноважені державні служби безпеки регулярно здійснюють огляд інформаційних систем, оцінюють їхню вразливість та проводять тренінги для персоналу, який керує об’єктами критичної інфраструктури.

Міжнародні дані свідчать, що поєднання традиційних заходів безпеки з сучасними технологіями кібербезпеки знижує ризики та зменшує наслідки кібератак. Країни Європейського Союзу, США та інші розвинені країни створюють системи попередження, які використовують штучний інтелект для аналізу великих масивів даних і виявлення потенційних загроз на ранній стадії для своєчасного розгортання захисту [11]. Інтеграція протоколів фізичної безпеки з кіберзахистом має важливе значення для захисту інформаційної безпеки та забезпечення стабільності і стійкості критичної інфраструктури в сучасному цифровому ландшафті.

Сучасні дослідження загроз інформаційній безпеці критичної інфраструктури вивчають зовнішні та внутрішні чинники, які можуть призвести до збоїв у роботі важливих систем. Поєднуючи традиційні захисні заходи з

сучасними технологіями кібербезпеки, держави можуть розробити багаторівневі системи управління ризиками, які підтримують стабільність у сучасному складному ландшафті. Чітке розуміння конкретних загроз, а також регулярний аудит безпеки та адаптивні заходи зменшують кількість інцидентів у кіберпросторі та сприяють розвитку критично важливої інфраструктури, яка лежить в основі економічної стабільності та соціальної безпеки [14].

1.3 Особливості вибору антивірусного програмного забезпечення для об’єктів критичної інфраструктури України

Захист критичної інфраструктури в Україні потребує особливої уваги з огляду на розвиток інформаційних технологій та ескалацію кіберзагроз. Ця інфраструктура, яка включає в себе енергетичні, транспортні, комунікаційні та інші системи, має важливе значення для безперебійної роботи держави. При виборі антивірусного програмного забезпечення для цих систем необхідно виходити за рамки технічних специфікацій, оскільки збої в роботі систем можуть призвести до серйозних економічних і соціальних наслідків. Використання комплексної стратегії, яка включає Закон України «Про основні засади забезпечення кібербезпеки», національні стандарти (ДСТУ), Стратегію кібербезпеки України на 2021-2025 роки та Методичні рекомендації щодо реагування суб’єктів кібербезпеки на різні типи кіберподій, забезпечує високий рівень захисту критично важливих інформаційних систем [1].

Основною нормативно-правовою базою, що регулює захист інформаційного простору держави, є Закон України «Про основні засади забезпечення кібербезпеки України» [15]. Цей закон передбачає, що цілісна система кібербезпеки повинна включати сертифіковані та ліцензовані засоби захисту, серед яких важливе значення має антивірусне програмне забезпечення. Крім того, закон наголошує на інтеграції як найважливішому компоненті стратегій кіберзахисту, що сприяє швидкому виявленню, оцінці та реагуванню на загрози, особливо ті, що впливають на об’єкти критичної інфраструктури [15].

Крім того, нормативно-правова база встановлює оперативні протоколи реагування на кіберінциденти для мінімізації шкоди під час кібератак.

Ключові компоненти інформаційної безпеки в Україні викладені в національних стандартах (ДСТУ), які визначають практичні вимоги до систем захисту інформації, що узгоджуються з міжнародними стандартами [7]. Стандарти ISO/IEC 27001:2023 та ISO/IEC 27002:2023 встановлюють вимоги до систем управління безпекою, адаптовані до національного контексту, та передбачають захисні заходи для зменшення кіберзагроз [16; 17]. Дото забезпечення передбачає встановлення критеріїв, які забезпечат, організації можуть оцінювати свої системи управління ризиками, розгортати механізми захисту, забезпечувати регулярне оновлення баз антивірусних сигнатур та проводити аудит інформаційних ресурсів. Застосовувані методи захисту гарантують надійність і сучасні стандарти продуктивності, сприяючи стабільності критично важливих інфраструктур.

Стратегія кібербезпеки України на 2021-2025 роки, затверджена Радою національної безпеки і оборони, окреслює основні напрями зміцнення кібербезпеки держави [18]. Даний документ встановлює дві основні цілі для підвищення кіберстійкості критичної інфраструктури. Вона спрямована на впровадження нових технологій та сприяння співпраці між державними органами та приватними підприємствами. Стратегія підкреслює необхідність сучасних рішень, які дозволяють швидко виявляти нові загрози, адаптуватися до кіберзагроз, що постійно змінюються, і включати антивірусні заходи в протоколи безпеки. Такий подвійний підхід захищає життєво важливі інформаційні ресурси, водночас підтримуючи доступність системи від кіберзагроз [18].

Методичні рекомендації для фахівців з безпеки, які вирішують проблеми кібербезпеки в кіберпросторі, пропонують практичні поради щодо посилення захисту активів інформаційних систем [19]. Ці рекомендації рекомендують використовувати антивірусне програмне забезпечення, оснащене функціями моніторингу в режимі реального часу, що сприяє швидкому виявленню

підозрілих дій і перешкоджає поширенню шкідливого програмного забезпечення. Підтримка централізованих систем управління захистом має вирішальне значення, оскільки вони консолідують передові знання про постійні загрози (APT) та інші елементи кібербезпеки для ретельного реагування на інциденти. Надійний захист активів безпеки залежить від постійної оцінки ефективності захисних заходів, проведення аудиту системи безпеки та забезпечення своєчасного оновлення алгоритмів виявлення загроз, особливо для об’єктів критичної інфраструктури [19].

Аналіз законодавчої бази вказує на те, що українські програми захисту критичної інфраструктури потребують ретельного відбору антивірусних рішень, які відповідають вимогам законодавства, національним стандартам України, стратегічним документам та методичним рекомендаціям. Ключовими елементами є інтегровані механізми захисту в системі, можливості швидкого реагування на кіберзагрози та здатність оновлювати програмне забезпечення відповідно до міжнародних стандартів якості. Використання сертифікованого та схваленого обладнання підвищує впевненість у його ефективності та надійності, що має вирішальне значення для забезпечення стабільної роботи критично важливих об’єктів.

Вибір антивірусного програмного забезпечення для об’єктів критичної інфраструктури вимагає ретельного аналізу законодавчої бази. Сюди входять аспекти українського законодавства з кібербезпеки, національні стандарти, встановлені стратегічні цілі з кібербезпеки та рекомендації щодо реагування на інциденти. Поєднання цих елементів створює надійну систему захисту, що сприяє швидкому виявленню загроз, ефективному управлінню інцидентами та стабільно високому рівню безпеки критично важливих інформаційних активів. Ця цілісна стратегія має важливе значення для забезпечення належного захисту державної інфраструктури та відіграє значну роль у підтримці операційної стабільності держави в умовах зростаючих кіберзагроз.

1.4 Огляд сучасних антивірусних рішень та їх функціональних можливостей

Сучасні антивірусні рішення захищають інформаційну безпеку як користувацьких систем, так і корпоративних мереж, постійно адаптуючись до нових загроз [31]. Хоча метод сигнатур залишається класичним підходом для антивірусного програмного забезпечення, він передбачає порівняння файлів з базами даних відомих шкідливих програм. Однак цей традиційний метод на основі сигнатур має обмеження, оскільки нові загрози, особливо атаки «нульового дня», можуть швидко з’являтися. Багато сучасних систем безпеки вдосконалюють цю традиційну модель, впроваджуючи евристичні алгоритми та методи моніторингу поведінки для виявлення незвичних дій, коли баз даних сигнатур недостатньо, забезпечуючи таким чином кращий захист [20].

Продукти безпеки на ринку використовують різні методи для забезпечення безпеки. Інструменти виявлення шкідливого програмного забезпечення використовують два підходи: виявлення на основі сигнатур ідентифікує вже існуючі загрози. Водночас поведінковий аналіз допомагає розпізнавати поліморфні загрози та нові варіанти вірусів зі зміненим кодом [27]. Системи розширеного виявлення та реагування (EDR) працюють у режимі реального часу, використовуючи алгоритми машинного навчання для передбачення та негайного реагування на потенційні загрози. Сучасні антивірусні рішення постійно контролюють файли та процеси після завантаження операційної системи, сприяючи швидкому виявленню та запобіганню підозрілим діям. Евристичні алгоритми аналізують поведінку системи, що дозволяє їм виявляти загрози, які можуть бути пропущені базами сигнатур. Модулі веб-захисту доповнюють існуючу систему, попереджаючи користувачів про потенційні загрози під час навігації, захищаючи від шкідливих веб-ресурсів [27].

Сучасне антивірусне програмне забезпечення забезпечує постійний нагляд за файлами та процесами, оцінюючи їхню активність одразу після запуску системи, що дозволяє швидко виявляти та припиняти підозрілі дії. Корпоративні

мережі створюють оптимальне середовище для такої інтеграції, оскільки системи управління інформацією та подіями безпеки (SIEM) сприяють централізованому моніторингу для посилення безпеки. Цей метод дозволяє отримати доступ до збору даних з численних джерел, забезпечуючи всебічне розуміння подій безпеки та покращуючи час реагування на інциденти [28].

Штучний інтелект і машинне навчання відкривають нові перспективи для вдосконалення систем антивірусного захисту. Штучний інтелект сприяє швидкому розпізнаванню незнайомих загроз і коригуванню алгоритмів виявлення відповідно до змін у кіберсередовищі. Сучасні виклики кібербезпеки вимагають передових стратегій захисту, що потребують евристичної поведінки та умовних сигнатур для роботи в якості інтегрованої системи захисту [21].

Сучасний ринок антивірусного програмного забезпечення пропонує постачальникам модульні та масштабовані рішення для індивідуальних і корпоративних потреб. Оцінка сумісності антивірусного програмного забезпечення з системами NGFW, компонентами безпеки EDR та хмарними платформами має важливе значення для забезпечення цілісного управління безпекою. Такий підхід забезпечує оптимальний захист, мінімізуючи при цьому витрати на управління системою та експлуатаційні витрати [29].

Таким чином, сучасні антивірусні рішення перетворюються на всеосяжні системи захисту, призначені для боротьби з відомими та новими загрозами. Використовуючи традиційні методи сигнатур, а також інноваційні методи, що включають евристичний аналіз і алгоритми машинного навчання, вони підтримують високі стандарти кібербезпеки. Комплексна стратегія, яка об’єднує програмні та апаратні компоненти та інтегрується з платформами SIEM, дозволяє швидко реагувати на кіберзагрози та підвищує адаптивність системи до швидких змін у кіберпросторі. Сучасні антивірусні рішення являють собою динамічну систему захисту, що постійно вдосконалюється, здатну ефективно протидіяти широкому спектру кіберзагроз, забезпечуючи тим самим загальний захист інформаційних систем.

Висновок до першого розділу

У першому розділі представлено детальний аналіз ключових концепцій інфраструктури та їх класифікацій а також оцінку загроз інформаційній безпеці.

Критична інфраструктура є основою державної та економічної діяльності, що робить її захист пріоритетом національної безпеки.

Категоризація цих об’єктів дозволяє визначити пріоритети їхнього захисту та ефективно розробити відповідні стратегії.

Аналіз загроз показує, що основними ризиками для інформаційної безпеки є кібератаки, стихійні лиха, техногенні аварії та терористичні акти.

Такі загрози можуть призвести до значних втрат і перебоїв у роботі критично важливої інфраструктури, що підкреслює необхідність створення та впровадження ефективних заходів захисту.

Отримані дані підкреслюють необхідність ретельного підходу до захисту критичної інфраструктури, який передбачає класифікацію об’єктів, оцінку ризиків і розробку відповідних захисних заходів.

РОЗДІЛ 2
РОЗРОБКА МЕТОДОЛОГІЇ ВИБОРУ АНТИВІРУСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Формування критеріїв оцінки антивірусного програмного забезпечення

В умовах стрімкого розвитку комп'ютерів та кіберзагроз вибір правильного антивірусного програмного забезпечення для критично важливих інформаційних систем є надзвичайно важливим [22]. Створення методології оцінки антивірусного програмного забезпечення передбачає встановлення критеріїв, які забезпечать об'єктивність аналізу, надійність висновків та успішність майбутніх оперативних рішень щодо впровадження програмного забезпечення [23]. Першочерговим завданням на цьому етапі є визначення технічних та організаційних характеристик, що впливають на ефективність захисту інформаційних систем, а також оцінка можливих операційних ризиків, пов'язаних з програмним забезпеченням.

Основна увага при розробці критеріїв приділяється тому, що антивірусне програмне забезпечення відповідає державним стандартам, викладеним у переліку ДССЗЗІ. Цей документ допомагає експертам оцінювати рішення, які пройшли початковий етап тестування і відповідають вимогам. Перші критерії оцінки були розроблені на основі цього переліку, що вимагало ретельного опису та кількісного аналізу перед вивченням вимог безпеки важливих систем.

Оцінка технічних характеристик антивірусного програмного забезпечення є ключовим фактором оцінки. Важливими показниками є ймовірність успішного виявлення, рівень помилкових спрацьовувань, використання ресурсів, час відгуку бази даних і коефіцієнт виявлення. Ці фактори працюють незалежно, щоб оцінити загальну ефективність системи захисту. Здатність програмного забезпечення ідентифікувати шкідливе програмне забезпечення відображає його

здатність розпізнавати нові та модифіковані варіанти вірусів, про що свідчить ймовірність виявлення. Частота помилкових спрацьовувань антивірусних алгоритмів впливає на точність цих систем, що призводить до надмірних сповіщень для користувачів і потенційного зниження продуктивності. Споживання ресурсів є важливим критерієм, який оцінює, як антивірусні операції впливають на системні ресурси комп'ютера, особливо для систем, що працюють у режимі реального часу. Здатність системи виявляти та нейтралізувати загрози залежить від середнього часу відгуку , тоді як регулярне оновлення баз даних гарантує, що антивірусне програмне забезпечення поінформоване про найсвіжіші ризики для безпеки. Таким чином, кожен показник допомагає комплексно оцінити можливості програмного забезпечення.

Ключовим аспектом розробки критеріїв є включення коефіцієнта ризику, який враховує потенційні операційні ризики. Ризиковий коефіцієнт сприяє ретельній оцінці технічних особливостей та різних факторів, які можуть порушити ефективне функціонування системи. Зокрема, він враховує непередбачувані збої, вразливості, що виникають при взаємодії з іншими компонентами інфраструктури, а також можливий вплив на загальну безпеку інформаційного середовища. Використовуючи цей коефіцієнт, організації можуть покращити свою оцінку систем управління ризиками, визнаючи антивірусні рішення життєво важливими елементами критично важливих інформаційних систем. Методологія передбачає два способи включення коефіцієнта ризику в систему: перший метод розглядає ризик як незалежний фактор у матриці критеріїв. На противагу цьому, другий метод вводить коригувальний коефіцієнт, який коригує ризик після обчислення базової оцінки на основі первинних технічних характеристик. Такий аналіз стає більш надійним, оскільки обидва методи сприяють комплексній оцінці, яка поєднує функціональність антивірусної програми з оцінкою операційного ризику.

Організації розробляють критерії, проводячи експертне оцінювання за допомогою методів експертних оцінок [24]. Цей метод дозволяє ранжувати критерії на основі їхньої важливості, а також присвоювати їм певну вагу.

Експерти проводять попарну порівняльну оцінку, щоб створити матрицю, яка встановлює відносну важливість вимог щодо інших факторів. Середнє геометричне значення обробляє індивідуальні результати експертів для отримання узагальнених ваг. Процес агрегування має на меті зменшити суб'єктивність особистих оцінок і водночас забезпечити більший консенсус серед групи експертів. Верифікація починається з розрахунку індексу узгодженості (CR) кожної матриці, який є життєво важливим для подальшого аналізу. Додаткові обговорення з експертами необхідні, якщо значення CR перевищує заздалегідь визначений поріг. Така методологія формування критеріїв сприяє точному та об'єктивному аналізу, інтегруючи кількісні дані та якісні фактори для оцінки ефективності антивірусного програмного забезпечення.

Особлива увага приділяється оцінці процесу відбору за критеріями, які безпосередньо впливають на якість захисту інформаційних систем. Розробка цих критеріїв вимагає врахування експлуатаційних вимог антивірусного програмного забезпечення та фундаментальних принципів кібербезпеки. Здатність системи виявляти відомі загрози та розпізнавати нові, невідомі типи шкідливих програм визначає значення критерію «ймовірність виявлення шкідливого коду». Аналогічно, критерій «частота помилкових спрацьовувань» відображає точність алгоритмів і адаптивність системи до змін навколишнього середовища, що впливає як на досвід користувача, так і на ефективність прийняття рішень у реальному часі. Критерії, що оцінюють навантаження на систему і час відгуку, мають вирішальне значення для розуміння того, наскільки добре антивірусне програмне забезпечення інтегрується в загальну інфраструктуру критично важливих інформаційних систем, допомагаючи запобігти помилкам у роботі важливих сервісів. З огляду на ці фактори, комплексний аналіз якості програмного забезпечення стає необхідним, оскільки сучасна кібербезпека вимагає такого підходу.

Встановлення критеріїв оцінки антивірусного програмного забезпечення методологічно базується на принципах багатокритеріального аналізу [25]. Ця

стратегія дозволяє інтегрувати різні типи інформації, включаючи експертні оцінки та кількісні результати тестування програмного забезпечення. Процес оцінювання забезпечує ретельне вимірювання продуктивності, поєднуючи технічні характеристики з аналізом операційних ризиків. Система попарного порівняння розбиває основну операцію на кілька компонентів для незалежної оцінки, які об'єднуються для отримання комплексної загальної метрики. Використовуючи цей підхід, можливо організувати дані та визначити рівні значущості різних критеріїв, які безпосередньо впливають на вибір впровадження антивірусного програмного забезпечення.

Група експертів з кібербезпеки, IT та управління ризиками відіграє вирішальну роль у формулюванні критеріїв оцінювання. Вони проводять попередній аналіз і спільно працюють над його результатами, щоб створити критерії, які гарантують високу достовірність результатів. У процесі оцінювання кожен експерт застосовує свій професійний досвід і знання для оцінки функцій антивірусного програмного забезпечення, а система компілює індивідуальні результати, щоб зменшити упередженість, зумовлену особистими думками.

Для досягнення об'єктивності важливо мати оптимальний розмір експертної групи, в ідеалі – не більше 12-13 осіб. Такий розмір сприяє балансу думок і підвищує ефективність обробки експертної інформації. Ефективний менеджмент має важливе значення для експертів під час їхньої взаємодії. Три основні методи, які використовують експертні групи, включають вільний обмін інформацією, регламентований обмін та ізольовану роботу. Метод оцінювання дозволяє інтегрувати об'єктивні точки зору, мінімізуючи упередженість при оцінюванні антивірусного програмного забезпечення для критично важливих інформаційних систем [30].

Створення критеріїв для оцінки антивірусного програмного забезпечення передбачає комплексний аналіз технічних специфікацій та врахування факторів ризику. Запропонований підхід дозволяє об'єктивно оцінити ефективність програмного забезпечення, охоплюючи всі найважливіші експлуатаційні аспекти та координуючи експертні оцінки за допомогою порівняльних методів.

Ця стратегія допомагає чітко визначити сильні та слабкі сторони різних антивірусних рішень і гарантує відповідність результатів встановленим державним стандартам і вимогам до критично важливих інформаційних систем. Поєднуючи технічний аналіз з оцінкою операційних ризиків, ця методологія максимізує безпеку і стабільність інформаційних систем, особливо в світлі зростаючих кіберзагроз.

Тому встановлення критеріїв для оцінки антивірусного програмного забезпечення за допомогою цієї методології є важливим кроком, який закладає основу для подальшого експертного оцінювання та прийняття обґрунтованих рішень. Обрані критерії відображають сучасні стандарти захисту інформаційних систем, поєднуючи технічні показники ефективності з оцінкою операційних ризиків для створення збалансованої системи оцінювання. Включення факторів ризику в модель оцінки дозволяє експертам проаналізувати кожен операційний аспект антивірусного програмного забезпечення, отримуючи достовірні результати аналізу, які полегшують прийняття важливих рішень щодо впровадження інформаційної системи.

Методологія використовує принципи багатокритеріального аналізу для всебічної оцінки антивірусного програмного забезпечення та підтримки узгодження думок експертів. Такий підхід має вирішальне значення для систем захисту, оскільки інтегрує поточні специфікації продукту з оцінкою операційних ризиків. Визначені критерії стануть основою для розробки системи оцінювання, яка гарантуватиме об’єктивний аналіз та дотримання стандартів безпеки при впровадженні антивірусного програмного забезпечення.

2.2 Розробка методики порівняльного аналізу антивірусних рішень

Створення тестів для антивірусних рішень має вирішальне значення для розгортання об’єктивних програмних засобів для захисту інформаційних систем. Цей підхід використовує багатокритеріальний аналіз для оцінки технічних характеристик антивірусного програмного забезпечення та операційних ризиків,

які можуть виникнути при його інтеграції в критичні інформаційні системи. Вибрані альтернативи отримують схвалення державних регуляторних органів та проходять експертну оцінку за допомогою методів експертного порівняння після початкового етапу відбору.

Створення бази даних альтернативних рішень, що включає антивірусні рішення, які відповідають вимогам, викладеним у переліку ДССЗІ, має важливе значення для впровадження цієї методології. Процес починається з дослідження ринку, за яким слідує відбір продуктів на основі їхньої сертифікації та відповідності вимогам, що гарантує виключення неперевірених або невідповідних варіантів. На наступному етапі основна увага приділяється визначенню критеріїв оцінки, які поділяються на групи: технічні атрибути, що вказують на ефективність захисту (ймовірність виявлення шкідливого програмного забезпечення, частота помилкових спрацьовувань, навантаження на систему, середній час відгуку і частота оновлень), і критерії, що відображають операційні ризики, які можуть вплинути на продуктивність системи. Важливим доповненням до технічних параметрів є включення коефіцієнта ризику, який враховує як прямі, так і непрямі ризики, пов’язані з використанням антивірусного програмного забезпечення [26].

Кожен експерт оцінює критерії та альтернативи окремо за допомогою методу попарних порівнянь. Така оцінка вимагає створення матриць попарних порівнянь для кожного критерію з метою визначення вагових коефіцієнтів. Індивідуальні експертні оцінки потім агрегуються з використанням середнього геометричного, що зменшує суб’єктивний вплив особистих думок і забезпечує більш узгоджену оцінку. Для перевірки узгодженості експертних оцінок розраховується індекс узгодженості (CR). Це важливий крок у перевірці узгодженості. Якщо значення CR перевищує заздалегідь визначений поріг експерти беруть участь у подальшому обговоренні для перегляду оцінок, забезпечуючи таким чином надійність даних.

Методологія складається з двох підходів до включення оцінки ризику. Перший метод розглядає ризик як окремий показник у матриці критеріїв, що

дозволяє йому безпосередньо впливати на оцінку альтернатив через загальний бал. Цей загальний бал отримується шляхом множення балів за критеріями на коефіцієнт безпеки (CVSS). Другий метод полягає в попередньому розрахунку базових балів технічних характеристик і подальшому застосуванні коригувального коефіцієнта експлуатаційного ризику. Разом ці підходи полегшують обчислення загальної оцінки, яка об'єднує технічні характеристики антивірусного рішення та пов'язані з ними операційні ризики.

Важливим аспектом методології є організація команди з 12-13 фахівців з кібербезпеки, інформаційних технологій та управління ризиками. Вони проводять індивідуальні попарні порівняння на основі заздалегідь визначених критеріїв, що призводить до консолідованої системи оцінювання [30]. Ця методологія дає об'єктивні результати, мінімізує вплив суб'єктивних факторів і забезпечує високу надійність прийнятих рішень. Вся робота здійснюється за допомогою спеціалізованого програмного забезпечення, яке включає модулі для збору оцінок, агрегування даних, аналізу узгодженості та прийняття остаточного рішення. Архітектура системи складається з декількох інтегрованих компонентів, включаючи базу даних альтернатив, модуль збору оцінок, модуль агрегування та модуль аналізу узгодженості експертних оцінок.

Створення методології порівняння антивірусних рішень пропонує ґрунтовний спосіб вибору найкращого програмного забезпечення для захисту критично важливих інформаційних систем. Метод попарного порівняння дозволяє агрегувати індивідуальні експертні оцінки та перевіряти узгодженість результатів, що полегшує ранжування антивірусних рішень з урахуванням операційних ризиків. Таким чином, включення коефіцієнта ризику в загальну модель оцінки гарантує, що обране антивірусне рішення відповідає сучасним стандартам безпеки та ефективно захищає інформаційні системи від кіберзагроз.

Під час розробки методології було враховано кілька практичних факторів, які дозволяють застосовувати її в реальних умовах. По-перше, система альтернативного відбору гарантує, що аналізуються лише ті рішення, які відповідають державним стандартам і демонструють доведену ефективність. По-

друге, попарні порівняння сприяють всебічній оцінці експертами кожного критерію, що підвищує точність визначення вагових коефіцієнтів. Нарешті, моніторинг узгодженості оцінок дозволяє оперативно коригувати результати у разі розбіжностей між думками експертів, що має вирішальне значення для мінімізації ризику прийняття помилкових рішень.

Комплексна модель, створена за допомогою цієї методології, дозволяє робити обґрунтований вибір при виборі антивірусного програмного забезпечення для критично важливих інформаційних систем. Отримані в результаті комплексні показники полегшують порівняння різних альтернатив та аналіз їх відповідності вимогам безпеки. Умовно стандартизована система оцінювання, що відповідає державним стандартам, створюється шляхом включення фіксованого коефіцієнта безпеки (CVSS) поряд з оцінками основних технічних параметрів. Такий підхід підвищує об’єктивність аналізу та надійність рішень, що приймаються.

Застосування цієї методології має велике практичне значення, оскільки вона спрощує вибір антивірусних рішень, які мають вирішальне значення для захисту інформаційних систем в умовах ескалації кіберзагроз. Комплексний підхід, що поєднує експертні оцінки, аналіз технічних характеристик та врахування операційних ризиків, дозволяє розробити систему, яка ефективно вирішує проблему вибору оптимального програмного забезпечення. Таким чином, ця методологія є одночасно науково обґрунтованою та орієнтованою на практичне використання, що спрощує її застосування в реальних проектах, спрямованих на посилення кібербезпеки критично важливих інформаційних систем.

2.3 Математична модель процесу прийняття рішень при виборі антивірусного програмного забезпечення

Математична модель має наступну структуру:

1. Рівень 1 - Мета:

Оцінка ефективності антивірусних рішень для критичних інформаційних систем.

2. Рівень 2 - Критерії:

– К1: Ймовірність виявлення шкідливого коду.

– К2: Рівень хибних спрацьовувань.

– К3: Навантаження на систему.

– К4: Середній час реакції.

– К5: Частота оновлень баз даних.

– К6: Ризиковий коефіцієнт (R), що характеризує потенційні експлуатаційні ризики.

– К7: Загальний коефіцієнт безпечності (CVSS) – фіксований показник, визначений за даними Переліку ДССЗЗІ (значення від 0 до 1).

3. Рівень 3 – Альтернативи: Антивірусні рішення з переліку ДССЗЗІ.

2.3.1 Агрегація експертних оцінок

– Для критеріїв :

Кожен експерт формує матрицю 7×7 з попарними порівняннями. Агрегована матриця формується за допомогою середнього геометричного.

$$a_{ij}^{agg} = (\prod_{k=1}^{n} a_{ij}^{(k)})^{\frac{1}{n}} \quad (2.1)$$

де n - загальна кількість експертів.

– Для альтернатив:

Аналогічним чином проводиться агрегування попарних матриць оцінок альтернатив для кожного критерію.

2.3.2 Перевірка узгодженості експертних оцінок

– Індивідуальна узгодженість: Для кожної матриці, створеної окремим експертом, розраховується співвідношення узгодженості (CR).

– Групова узгодженість: Агрегована матриця також перевіряється за допомогою CR. Якщо значення CR перевищує встановлений поріг (наприклад, 0.1), експертна група проводить додаткове обговорення з метою корекції оцінок.

2.3.3 Інтеграція ризикового коефіцієнта

– **Підхід А:**

Ризиковий коефіцієнт включається до матриці критеріїв як К6. Загальний бал альтернативи розраховується за формулою, де сума оцінок за критеріями К1–К6 множиться на фіксований коефіцієнт CVSS (К7).

$$P_i = (\sum_{j=1}^{6} w_j \cdot s_{ij}) \cdot \ CVSS_i \tag{2.2}$$

– **Підхід В:**

Спочатку розраховується базова оцінка за критеріями К1–К5:

$$B_i = \sum_{j=1}^{5} w_j \cdot s_{ij} \tag{2.3}$$

після чого застосовується коригувальний множник для врахування ризику, і отриманий результат множиться на $CVSS_i$:

$$P_i = (B_i \cdot (1 - k \cdot R_i)) \cdot CVSS_i \tag{2.4}$$

2.3.4 Прийняття рішення

– Обчислення загального показника: для кожної альтернативи обчислюється загальний бал P_i за обраним підходом інтеграції ризику та коефіцієнта CVSS.

– Вибір оптимальної альтернативи: Альтернатива з максимальним P_i вважається оптимальним вибором для впровадження в критичних інформаційних системах.

– Додатковий аналіз: Якщо групова узгодженість (CR) не відповідає встановленим нормам (наприклад, CR> 0.1), проводиться додаткове обговорення експертної групи для уточнення оцінок.

2.4 Алгоритм роботи експертної системи

Алгоритм дотримується покрокового процесу, що включає ретельний аналіз вхідних даних, агрегування експертних оцінок, перевірку узгодженості та врахування факторів ризику для визначення оптимальних показників рішення. Ця структурованість забезпечує прозорість та відтворюваність результатів вибору.

На початковому етапі система збирає дані, які включають інформацію про антивірусні рішення та визначені критерії оцінки. Ці дані складаються з об’єктивних технічних показників (таких як ймовірність виявлення шкідливого програмного забезпечення, рівень помилкових спрацьовувань, навантаження на систему, середній час відгуку та частота оновлення баз даних) та значень коефіцієнтів безпеки (CVSS) разом із коефіцієнтом ризику. Важливо, щоб інформація про антивірусні продукти та критерії була представлена у стандартизованому форматі для коректної обробки системою. Потім система обробляє цю інформацію за допомогою експертного аналізу даних з використанням методів попарного порівняння.

Кожен експерт будує свою матрицю шляхом попарних порівнянь, підкреслюючи важливість різних критеріїв та альтернатив. Цей етап значною мірою спирається на досвід та інтуїцію фахівців у сфері кібербезпеки. Система об’єднує вихідні матриці за допомогою аналізу середнього геометричного, щоб зменшити вплив особистих упереджень і отримати об’єктивне представлення даних. Система оцінює узгодженість загальної оцінки шляхом розрахунку індексу узгодженості (CR). Якщо CR перевищує заздалегідь визначений поріг, система ініціює подальше обговорення серед експертної групи для коригування початкових оцінок. Такий ітеративний підхід дозволяє досягти консенсусу та

підвищити достовірність експертних суджень. Обговорення може включати перегляд аргументації та, за потреби, повторне заповнення матриць порівнянь.

Розрахункова модель включає фактори ризику, використовуючи два методи їх введення. Вибір конкретного методу залежить від стратегічних пріоритетів організації, що проводить оцінку.

Початковий метод використовує коефіцієнт ризику в оцінці критеріїв, отримуючи загальну оцінку рішення шляхом множення зважених оцінок за критеріями (K1-K6) на постійний коефіцієнт CVSS. Тут ризик розглядається як один із фундаментальних критеріїв, що має свою вагу.

Другий метод розраховує базовий бал у діапазоні від K1 до K5, застосовує поправки, пов’язані з ризиком, а потім виконує множення коефіцієнта CVSS. У цьому випадку ризик діє як модифікатор загальної технічної оцінки, коригуючи її залежно від потенційних загроз. Це дозволяє більш гнучко враховувати непередбачувані фактори.

Система оцінює зібрані показники, щоб визначити найефективніше антивірусне рішення. Вона обирає варіант, який набрав найбільшу загальну кількість балів.

Нижче наведено блок-схему алгоритму роботи експертної системи, що демонструє послідовність її основних етапів:

Рисунок 2.1 – Структурна схема математичної моделі

Дана модель забезпечує послідовну обробку даних, об’єктивно агрегує експертні оцінки та враховує параметри ризику, що дозволяє приймати обґрунтовані та оптимальні рішення для підтримки високого рівня інформаційної безпеки в критично важливих системах.

Висновок до другого розділу

Математична модель дозволяє стандартизовано оцінювати антивірусні рішення за допомогою технічних завдань та якісних експертних оцінок. Це рішення пропонує ретельне дослідження інформаційної безпеки критично важливих систем для прийняття рішень. Використовуючи обчислення середнього геометричного для узагальнення експертних оцінок, модель зменшує особисті упередження, забезпечуючи надійне представлення даних щодо загальної ефективності системи.

Перевірка узгодженості експертних оцінок за допомогою показника співвідношення узгодженості (CR) є важливим етапом, оскільки своєчасне виявлення можливих розбіжностей дозволяє усунути недоліки в попарних порівняннях та гарантувати логічну послідовність і внутрішню сумісність даних.

Інтегрування ризикового коефіцієнта за допомогою двох підходів демонструє адаптивність моделі до унікальних операційних умов. Кожен метод пропонує більш точне представлення потенційних ризиків, враховуючи як кількісні, так і якісні фактори, що в кінцевому підсумку підвищує точність і ефективність оцінки.

Алгоритм експертної системи демонструє свою ефективність як інструмент захисту інформації через його систематичний процес, від збору даних до остаточного прийняття рішення. Практичне застосування результатів алгоритму в реальних робочих середовищах стало можливим завдяки його чіткій структурній схемі, яка забезпечує прозорі та зрозумілі процеси прийняття рішень, які додатково оптимізують обчислювальні процеси. Загалом, результати дослідження підтверджують наукову новизну підходу, обґрунтовують достовірність отриманих даних та відкривають нові можливості для практичного застосування розробленої моделі у сфері забезпечення інформаційної безпеки.

РОЗДІЛ 3
ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ЕКСПЕРТНОЇ СИСТЕМИ

3.1 Архітектура експертної системи

Для програмної реалізації експертної системи використовується мова програмування Python. Цей вибір обґрунтований не лише широкою доступністю та великою кількістю бібліотек, але й високою читабельністю коду, що спрощує розробку та подальшу підтримку, а також ефективними засобами для обробки даних та створення графічних інтерфейсів. Вона використовує модульну архітектуру, що є поширеним методом у сучасній розробці програмного забезпечення. Такий вибір має кілька переваг: покращує організацію коду, спрощує його розуміння, дозволяє тестувати окремі компоненти, а також дає змогу надалі підтримувати або розширювати функціонал. Така декомпозиція системи на логічно завершені блоки сприяє також паралельній розробці та полегшує локалізацію можливих помилок. Кожен логічний компонент міститься у власному файлі Python («.py»), що чітко розподіляє обов’язки та мінімізує взаємозалежності між частинами програми.

Директиви імпорту в програмному коді перевіряють наявність бібліотек Python та сторонніх пакетів. Додаток використовує стандартну бібліотеку Tkinter для створення графічного інтерфейсу користувача (з модулями «tkinter» і «tkinter.ttk»). Будучи нативною бібліотекою Python, Tkinter забезпечує легку інтеграцію та не потребує встановлення додаткових залежностей у більшості випадків. Такий вибір забезпечує високу крос-платформну сумісність, що дозволяє програмі функціонувати на різних операційних системах. Код враховує це, перевіряючи наявність Tkinter у системі та надаючи інструкції зі встановлення для користувачів Linux («sudo apt-get install python3-tk»). Для покращення візуальної якості на різних платформах програма використовує модуль «platform» для визначення поточної операційної системи та застосування

відповідних шрифтів за замовчуванням («DejaVu Sans» для Linux та «Segoe UI» для інших). Для досягнення цілісного та професійного зовнішнього вигляду з одночасним покращенням взаємодії з користувачем було розроблено спеціальний модуль під назвою «styles». Цей модуль централізує налаштування візуальних стилів за допомогою функції «setup_custom_styles». Він включає заводські функції («create_custom_button», «create_custom_label») для створення стандартизованих, стилізованих елементів керування (кнопок і написів), які замінюють стандартні віджети Tkinter зі специфічними кольорами, шрифтами і ефектами наведення. Важливим аспектом інтерфейсу є візуалізація результатів аналізу, яку полегшує інтеграція бібліотеки Matplotlib. Візуальне представлення даних у вигляді графіків та діаграм значно спрощує сприйняття складних числових результатів експертної оцінки. Клас «FigureCanvasTkAgg» з «matplotlib.backends.backend_tkagg» дозволяє безпосередньо вбудовувати графіки і діаграми, які відображають результати обчислення AHP, у вікна програми Tkinter.

Бібліотека Numpy полегшує обробку даних та обчислення. Вона відіграє важливу роль у реалізації математичної основи методу аналізу ієрархій (AHP), пропонуючи ефективні інструменти для роботи з багатовимірними масивами (матрицями). Ці масиви є основою для визначення ваг критеріїв та визначення пріоритетів альтернатив у модулі «ahp_calculator».

Система управління базами даних MongoDB використовується для зберігання системних даних, які включають інформацію про користувачів, антивірусні продукти, а також критерії та порівняння, надані експертами. Використання документо-орієнтованої MongoDB забезпечує гнучкість схеми даних, що є перевагою при можливій модифікації набору критеріїв чи атрибутів антивірусних рішень у майбутньому. Взаємодія з цією базою даних відбувається через офіційний драйвер Python, Pymongo. Модуль «database_connection» містить всю логіку, необхідну для роботи з базою даних: встановлює з’єднання, виконує завдання читання/запису/оновлення/видалення (CRUD) та контролює структуру даних на рівні індексів. Код чітко визначає створення унікальних

індексів («create_index») для колекцій, забезпечуючи цілісність даних і підвищуючи продуктивність запитів. Крім того, тип «ObjectId» з бібліотеки «bson» обробляє унікальні ідентифікатори для документів MongoDB.

Бібліотека Python «hashlib» є життєво важливою для захисту облікових даних користувача. Вона використовує спеціалізований модуль «password_utils» для виконання функцій «hash_password» та «verify_password», які реалізують рекомендований криптографічний алгоритм PBKDF2-HMAC-SHA256 з «сіллю». Цей метод ефективно хешує і перевіряє паролі, запобігаючи зберіганню паролів у відкритому вигляді і значно ускладнюючи їх підбір. Модуль «base64» також використовується для кодування та декодування хешу та «солі» під час зберігання.

Система складається з декількох основних модулів. Основний модуль, «main_app», є точкою входу та центральним контролером програми. Він ініціалізує графічний інтерфейс, керує подіями користувача, керує навігацією між вікнами та викликає функції з інших модулів для виконання необхідних операцій. Модуль «main_app» використовує «database_connection» для завантаження та зберігання даних і «ahp_calculator» для виконання розрахунків AHP за запитом користувача. Він також реалізує візуальні стилі з «styles». Модуль «database_connection» обробляє всі взаємодії з MongoDB, слугуючи програмним інтерфейсом для доступу до даних і приховуючи основні деталі Pymongo. Крім того, цей модуль співпрацює з «password_utils» для процесів автентифікації користувачів . Модуль «ahp_calculator» охоплює основну логіку методу AHP, виконуючи функцію «calculate_weights» та інші важливі обчислення. Як зазначалося раніше, «password_utils» і «styles» пропонують утиліти безпеки і візуального дизайну відповідно.

Нижче наведено блок-схему програми:

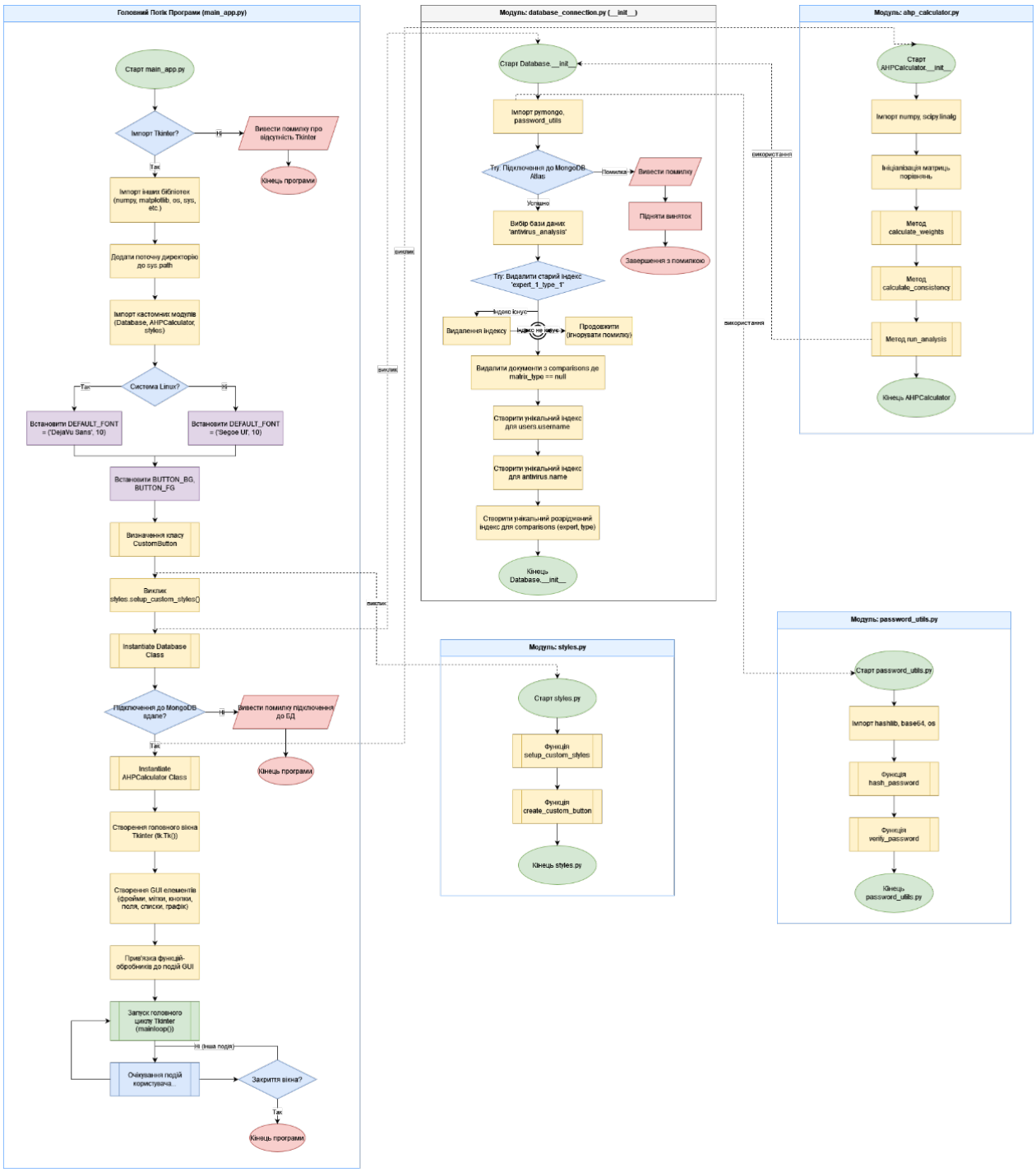

Рисунок 3.1 – Блок-схема програми

3.2 Розробка бази даних та механізму логічного виведення

Робота створеної експертної системи тісно пов’язана з двома основними елементами, описаними у встановленій методології: підсистемою зберігання даних, що виконує роль бази знань, та обчислювальним ядром, яке реалізує механізм виведення на основі методу ієрархічного аналізу.

3.2.1. Реалізація підсистеми зберігання даних

Підсистема зберігання даних функціонує як база знань для експертної системи, використовуючи документно-орієнтовану систему управління базами даних (СУБД) MongoDB. MongoDB була обрана за її адаптивність при роботі зі слабоструктурованими даними, такими як матриці порівняння різної розмірності, а також за її відмінну масштабованість. Вся логіка взаємодії з базою даних знаходиться в модулі "database_connection", де визначено клас "Database".

Для підключення до MongoDB за допомогою Python використовується офіційний драйвер Pymongo. Підключення до сервера MongoDB ініціюється при створенні екземпляру класу " Database " за допомогою "MongoClient". Код включає обробку помилок, пов'язаних з проблемами з'єднання, за допомогою блоку "try ... except". Після підключення обирається конкретна база даних "antivirus_analysis" для подальших операцій. Крім того, перевіряється доступність сервера бази даних за допомогою команди "ping".

```
import pymongo
from pymongo import MongoClient
from password_utils import hash_password, verify_password
from datetime import datetime
class Database:
    def __init__(self):
        try:
            self.client = MongoClient()
            self.db = self.client['antivirus_analysis']
            self.client.admin.command('ping')
            print("Успішне підключення до MongoDB")
        except Exception as e:
            print(f"Помилка при підключенні до MongoDB: {str(e)}")
            raise
```

Структура бази даних складається з трьох основних колекцій:

1. «users»: Містить детальну інформацію про користувачів системи, таких як експерти та адміністратори, включаючи їхні логіни та хешовані паролі.

2. «antivirus»: Містить довідкову інформацію про проаналізовані продукти, включаючи їхні назви та значення оцінок CVSS.

3. «comparisons»: Містить результати експертних оцінок, які включають матриці попарних порівнянь для критеріїв та альтернатив. Вони пов'язані з конкретним експертом і типом порівняння (наприклад, "criteria" або "alt_by_K1"), а також з розрахованим коефіцієнтом узгодженості (CR) і часовими мітками.

Система генерує колекції автоматично при першому запуску або за їх відсутності. Крім того, відбувається початкове заповнення баз даних: якщо колекція "users" порожня, створюється обліковий запис адміністратора за замовчуванням ("admin"/"admin") з хешованим паролем; якщо колекція "antivirus" порожня, вона заповнюється списком попередньо визначених антивірусів разом з їхніми CVSS-значеннями.

```
if ‘users’ not in self.db.list_collection_names():
    print("Створення колекції users та додавання адміністратора за замовчуванням")
    self.db.create_collection(‘users’)
    self.db.users.insert_one({
        ‘username’: ‘admin’,
        ‘password’: hash_password(‘admin’),
        ‘role’: ‘admin’
    })

if ‘antivirus’ not in self.db.list_collection_names():
    print("Створення колекції antivirus та додавання даних за замовчуванням")
    self.db.create_collection(‘antivirus’)
    self.db.antivirus.insert_many([
```

```
            {‘name’: ‘ESET NOD32 Antivirus’, ‘cvss’: 0.84},
            {‘name’: ‘Bitdefender Antivirus’, ‘cvss’: 0.8},
            {‘name’: ‘Avast Business Antivirus’, ‘cvss’: 0.65},
        ])
    if ‘comparisons’ not in self.db.list_collection_names():
        print("Створення колекції comparisons")
        self.db.create_collection(‘comparisons’)
```

Індекси MongoDB відіграють важливу роль у підтримці цілісності даних та оптимізації запитів. Метод "__ init __" класу "Database" відповідає за створення декількох ключових індексів:

```
    self.db.users.create_index([(‘username’, 1)], unique=True)
    self.db.antivirus.create_index([(‘name’, 1)], unique=True)
    self.db.comparisons.create_index(
        [(‘expert’, 1), (‘matrix_type’, 1)],
        unique=True,
        sparse=True,
        name=‘expert_1_matrix_type_1’
    )
    try:
        self.db.comparisons.drop_index(‘стара_назва_індексу’)
    except:
        pass
```

"unique=True" для полів "username" та "name" запобігає створенню дублікатів користувачів та антивірусів.

Комбінований індекс "[('expert', 1), ('matrix_type', 1)]" у поєднанні з опціями "unique = True" та "sparse = True" гарантує, що кожен експерт може мати лише одну матрицю певного типу (наприклад, один експерт відповідає одній матриці "criteria"). Натомість опція "sparse=True" покращує індекс, враховуючи лише ті документи, в яких присутні обидва поля ("expert" і "matrix_type").

Клас "Database" пропонує різні методи для виконання основних CRUD-

операцій (Create, Read, Update, Delete) над даними, що спрощує інтеграцію з Pymongo для основного коду програми ("main_app"). Важливими методами є наступні:

– « get_users()»: Повертає список усіх користувачів.

– «get_user(username , password)»: Знаходить користувача за іменем користувача і перевіряє пароль за допомогою «verify_password» з модуля «password_utils».

– «add_user(username,password,role='expert')»: Додає нового користувача, хешуючи пароль за допомогою «hash_password».

– «update_user(old_username,new_username,new_password,new_role): Змінює інформацію про користувача. Якщо змінюється ім'я користувача, оновлюється також поле «expert» у відповідних записах колекції «comparisons». Транзакції використовуються для атомарності узгодженості.

– « delete_user (username)»: Видаляє користувача і всі пов'язані з ним порівняння з колекції «comparisons».

– «get_antiviruses()»: Повертає список усіх антивірусів.

– «update_antivirus(av_id,new_name,new_cvss)»: Оновлює дані антивірусу.

– «delete_antivirus(antivirus_id)»: Видаляє антивірус.

– «save_comparison(expert,matrix_type, matrix,cr)»: Зберігає або змінює матрицю порівняння для експерта. Використовує логіку «upsert» (оновлення або вставка) всередині транзакції, забезпечуючи атомарність і запобігаючи конфліктам при одночасному доступі.

– «get_expert_comparison(expert,matrix_type)»: Отримує збережену матрицю порівняння для певного експерта та типу матриці, виконується в межах транзакції для забезпечення послідовного зчитування.

– «get_expert_names()»: Повертає список логінів усіх користувачів з роллю 'expert'.

Транзакції MongoDB використовуються для операцій, що потребують внесення змін до кількох документів або покладаються на попередні стани (наприклад, "update_user" або "save_comparison"). Це забезпечує атомарність

операцій: або кожна зміна в транзакції успішно виконується ("session.commit_transaction()"), або жодна не застосовується у випадку помилки ("session.abort_transaction()").

```
def save_comparison(self, expert, matrix_type, matrix, cr):
    try:
        self.client.admin.command('ping')
        with self.client.start_session() as session:
            try:
                with session.start_transaction():
                    existing = self.db.comparisons.find_one({
                        'expert': expert,
                        'matrix_type': matrix_type
                    }, session=session)
                    if existing:
                        result = self.db.comparisons.update_one(
                            {'_id': existing['_id']},
                            {'$set': {
                                'matrix': matrix,
                                'cr': cr,
                                'last_updated': datetime.now()
                            }},
                            session=session
                        )
                    else:
                        result = self.db.comparisons.insert_one({
                            'expert': expert,
                            'matrix_type': matrix_type,
                            'matrix': matrix,
                            'cr': cr,
                            'created_at': datetime.now(),
```

```
                    'last_updated': datetime.now()
                }, session=session)
                saved = self.db.comparisons.find_one({'_id': result.inserted_id if not existing else existing['_id']}, session=session)
                if saved:
                    session.commit_transaction()
                    print(f"Порівняння для {expert}, тип {matrix_type} збережено/оновлено.")
                    return True
                else:
                    print(f"Помилка перевірки після збереження для {expert}, тип {matrix_type}.")
                    session.abort_transaction()
                    return False
            except Exception as e:
                print(f"Помилка в транзакції збереження порівняння: {str(e)}")
                session.abort_transaction()
                return False
        except Exception as e:
            print(f"Невідома помилка при збереженні порівняння: {e}")
            return False
```

Крім того, модуль усуває специфічні помилки Pymongo, такі як "pymongo.errors.ConnectionFailure" і "pymongo.errors.OperationFailure", підвищуючи надійність бази даних.

Підсистема зберігання даних забезпечує структуроване, надійне та ефективне управління всією інформацією, необхідною для роботи експертної системи, такою як дані користувачів, антивірусні продукти та результати оцінок спеціалістів.

3.2.2. Реалізація обчислювального методу

Обчислювальний метод, що лежить в основі експертної системи, інтегровано в модуль «ahp_calculator». Цей модуль містить клас AHPCalculator, який інкапсулює всю логіку, пов’язану з Методом Аналізу Ієрархій (AHP). Для ефективних математичних обчислень, особливо для операцій з матрицями, використовується бібліотека NumPy.

Клас AHPCalculator ініціалізується зі словником RI (Random Index), який зберігає стандартні значення індексу випадкової узгодженості, необхідні для оцінки узгодженості матриць парних порівнянь.

```
class AHPCalculator:
    def __init__(self):
        self.RI = {1: 0, 2: 0, 3: 0.58, 4: 0.9, 5: 1.12, 6: 1.24, 7: 1.32, 8: 1.41, 9: 1.45, 10: 1.49}
```

Центральним методом для визначення вагових коефіцієнтів критеріїв або локальних пріоритетів альтернатив є calculate_weights. Цей метод реалізує метод власних векторів для знаходження вектора пріоритетів.

```
    def calculate_weights(self, matrix):
        n = len(matrix)
        A = np.array(matrix, dtype=float)
        eigenvalues, eigenvectors = eig(A)
        max_idx = np.argmax(eigenvalues.real)
        max_eigval = eigenvalues[max_idx].real
        max_eigvec = eigenvectors[:, max_idx].real
        weights = max_eigvec / np.sum(max_eigvec)
        ci = (max_eigval - n) / (n - 1) if n > 1 else 0
        cr = ci / self.RI[n] if n > 1 else 0
        return weights, cr
```

Метод calculate_weights виконує наступні кроки:

1. Отримує розмірність матриці n.

2. Перетворює вхідну матрицю matrix на NumPy масив A.

3. Використовує функцію eig (з бібліотеки scipy.linalg) для обчислення власних значень та власних векторів матриці A.

4. Знаходить максимальне дійсне власне значення (λmax) та відповідний йому власний вектор.

5. Обчислює вектор ваг weights шляхом нормалізації власного вектора (ділення кожного елемента на суму всіх елементів вектора).

6. Розраховує індекс узгодженості (CI) за формулою CI=(λmax−n)/(n−1).

7. Розраховує коефіцієнт узгодженості (CR), ділячи CI на відповідне значення випадкового індексу RI для матриці розмірності n.

8. Повертає обчислений вектор ваг та коефіцієнт узгодженості CR.

Для узагальнення оцінок від кількох експертів перед розрахунком фінальних ваг використовується метод geometric_mean_aggregation. Він агрегує (усереднює) список матриць парних порівнянь в одну, обчислюючи середнє геометричне для кожної відповідної комірки матриць.

```
def geometric_mean_aggregation(self, matrices):
    if not matrices:
        return None
    n = len(matrices[0])
    result = np.ones((n, n))
    for i in range(n):
        for j in range(n):
            product = 1.0
            for matrix in matrices:
                product = matrix[i][j]
            result[i][j] = product  (1.0 / len(matrices))
    return result.tolist()
```

Нарешті, клас AHPCalculator реалізує два підходи для розрахунку фінальної інтегральної оцінки Pi для кожної альтернативи (антивірусного рішення):

Підхід A (calculate_approach_a): Враховує всі критерії, включаючи ризиковий коефіцієнт (K6), при розрахунку зваженої суми, яка потім множиться на значення CVSS.

```
def calculate_approach_a(self, criteria_weights, alt_weights_by_criteria, cvss_values):
    n_alternatives = len(alt_weights_by_criteria[0])
    results = []
    for i in range(n_alternatives):
        weighted_sum = 0
        for j in range(len(criteria_weights)):
            weighted_sum += criteria_weights[j]  alt_weights_by_criteria[j][i]
        result = weighted_sum  cvss_values[i] [cite: 220]
        results.append(result)
    return results
```

Підхід B (calculate_approach_b): Спочатку розраховує базову оцінку на основі критеріїв K1-K5, потім коригує її за допомогою ризикового коефіцієнта (ваги за критерієм K6) та параметра k, і лише після цього множить на CVSS.

```
def calculate_approach_b(self, criteria_weights, alt_weights_by_criteria, risk_values, cvss_values, k=0.5):
    n_alternatives = len(alt_weights_by_criteria[0])
    results = []
    for i in range(n_alternatives):
        base_sum = 0
        for j in range(5):
            base_sum += criteria_weights[j]  alt_weights_by_criteria[j][i]
        adjusted = base_sum  (1 - k  risk_values[i])
        result = adjusted  cvss_values[i]
        results.append(result)
    return results
```

Дані обчислювальні методи формують ядро механізму логічного

виведення експертної системи, дозволяючи перетворювати експертні судження, виражені у вигляді матриць парних порівнянь, на кількісні оцінки та фінальний рейтинг антивірусних рішень.

3.3 Реалізація інтерфейсу користувача

Графічний інтерфейс користувача (GUI) відіграє життєво важливу роль в експертних системах, полегшуючи взаємодію експерта з системою. Він дозволяє користувачам вводити дані, виконувати аналіз та отримувати результати. Для цієї реалізації ми обрали Tkinter, стандартну бібліотеку Python для роботи з графікою. На користь цього рішення свідчить кілька причин: Tkinter входить до складу стандартних дистрибутивів Python, що спрощує розгортання додатків; вона є кросплатформенною, сумісною з багатьма операційними системами та надає достатньо можливостей для створення функціонального та візуально привабливого інтерфейсу. Для обробки рідкісних ситуацій, коли Tkinter може бути недоступний на системі користувача, на початку головного файлу програми (main_app.py) включено блок try...except.

```
try:
    import tkinter as tk
    from tkinter import ttk
except ImportError:
    print("Помилка: Не встановлено tkinter. Встановіть його командою:")
    print("sudo apt-get install python3-tk")
    exit(1)
```

Поточний код гарантує, що програма не запуститься без необхідних бібліотек графічного інтерфейсу та інформує користувачів про те, як їх встановити.

Основний логічний інтерфейс знаходиться у модулі «main_app». Інтерфейс структури пропонує декілька головних вікон або фреймів для виконання різних завдань:

Вікно аутентифікації (рис. 3.2). Початковий екран, на якому користувачеві представлені поля для відображення імені, імені користувача та пароля, а також кнопки для входу в систему або реєстрації нового користувача.

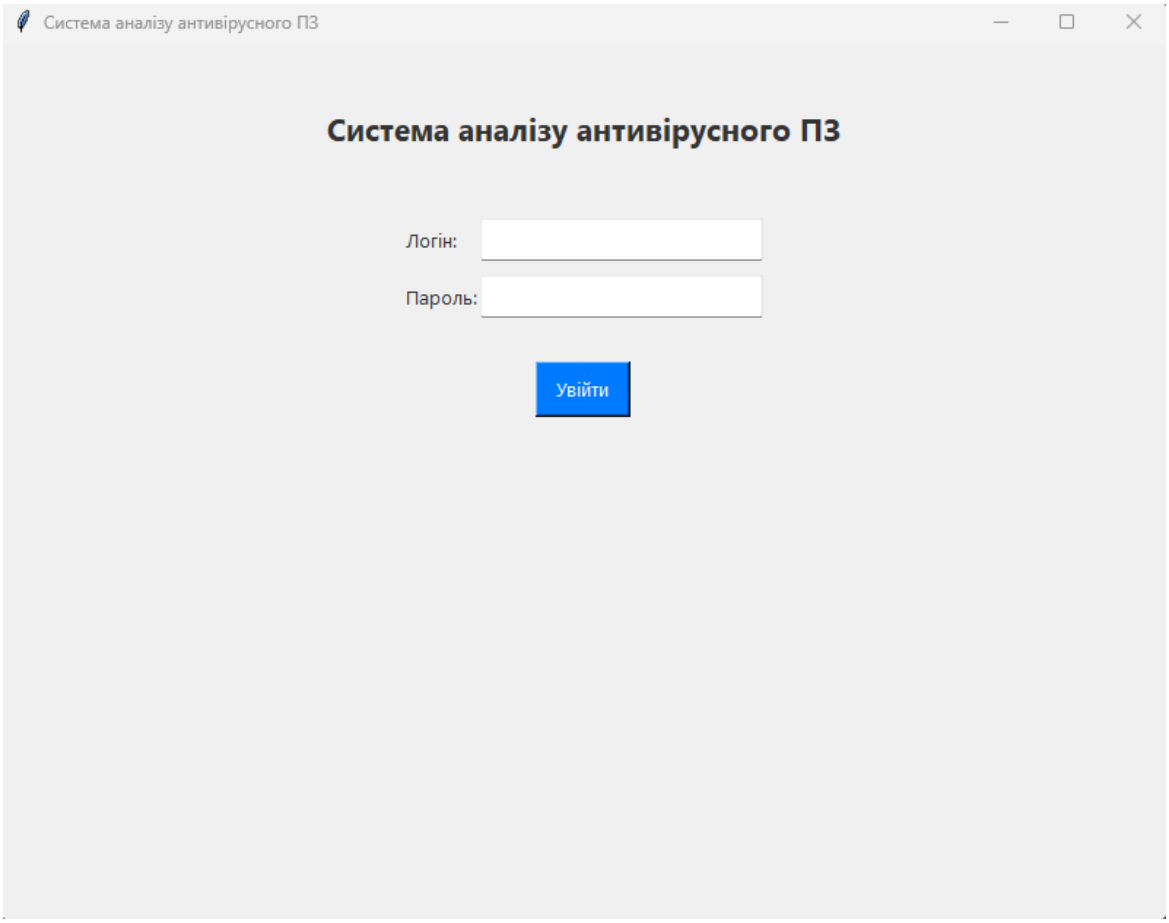

Рисунок 3.2 – Вікно аутентифікації

Вікно адміністратора активується після успішної аутентифікації в якості адміністратора (рис. 3.3). Ця основна робоча область надає доступ до всіх основних функціональних систем, включаючи списки управління, антивірусні інструменти, критерії, процедури запуску, введення порівнянь та перегляд результатів аналізу.

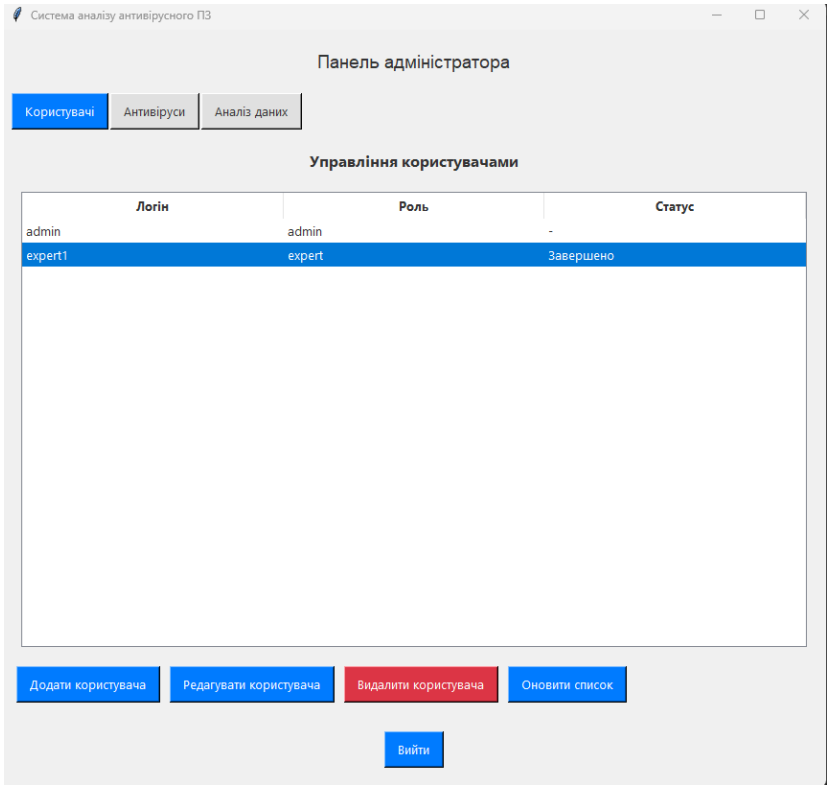

Рисунок 3.3 – Головне робоче вікно адміністратора

Вікно введення попарних порівнянь (рис. 3.4). Спеціалізований інтерфейс, який дозволяє експертам вводити оцінки на основі порівнянних критеріїв або альтернатив. Усі пункти які не є активними для заповнення користувачем автоматично запповнюються.

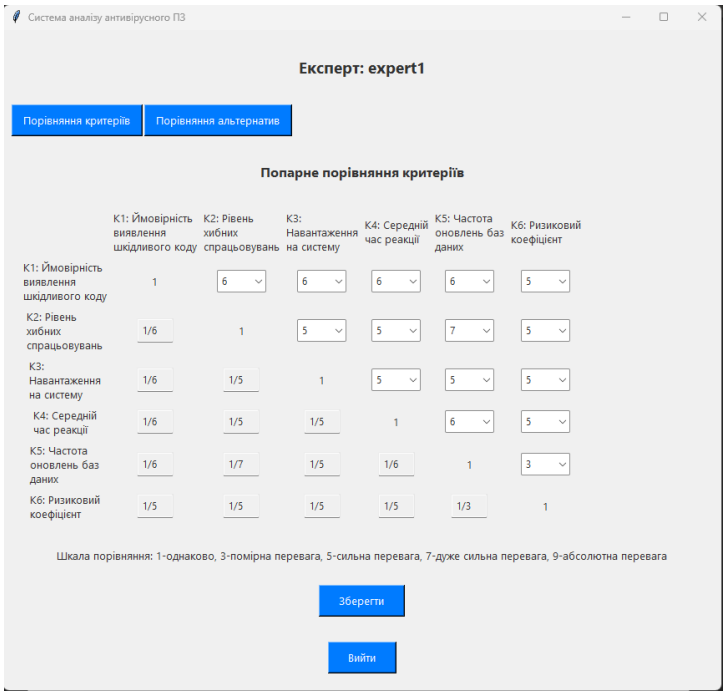

Рисунок 3.4 – Вікно введення попарних порівнянь

Вікно відображення результатів (рис. 3.5). Відображає результати розрахунків AHP, включаючи зважені критерії, локальні та глобальні пріоритети, а також антивіруси, у вигляді таблиць і графіків.

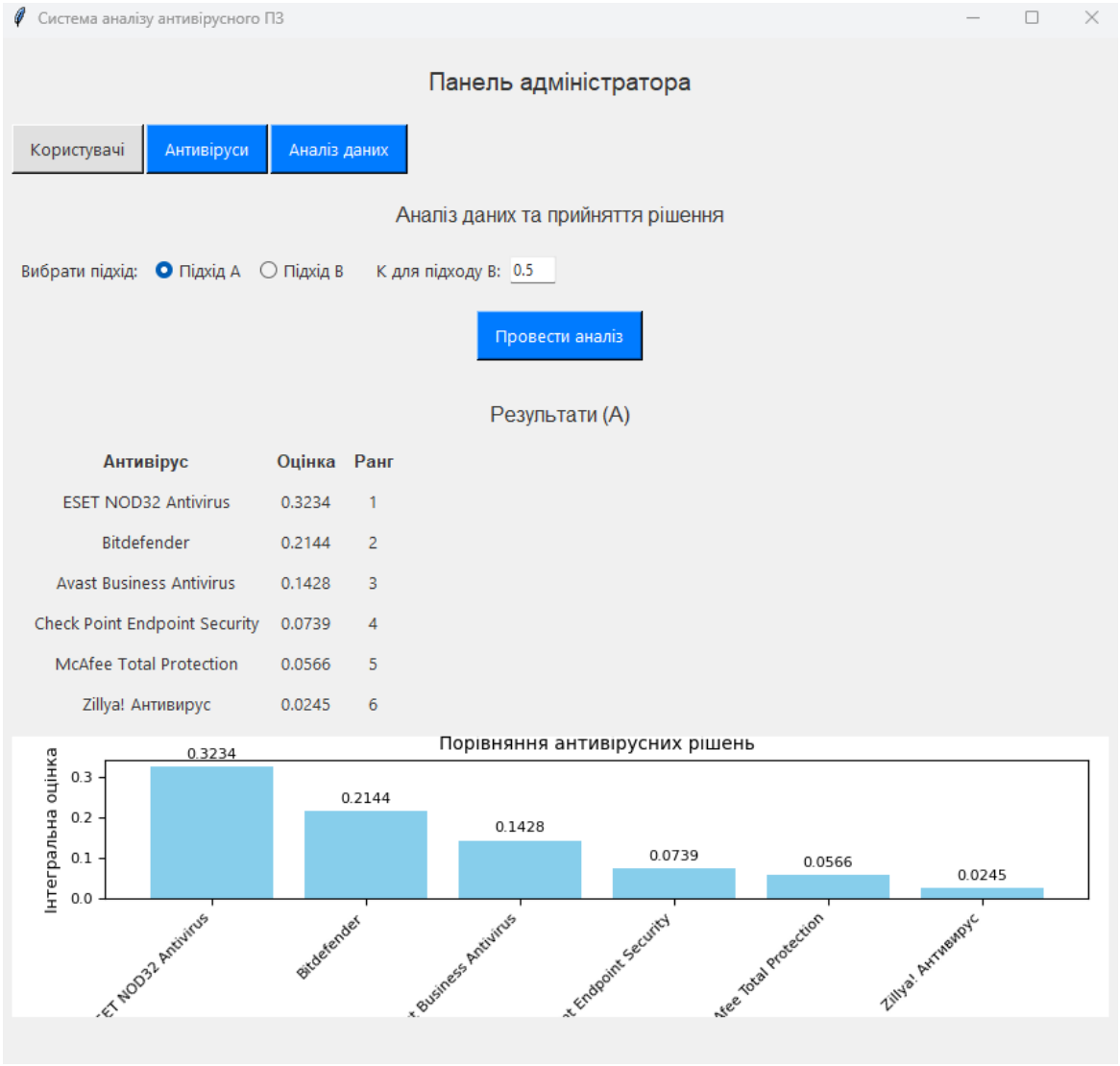

Рисунок 3.5 – Вікно відображення результатів

Для забезпечення єдиного візуального стилю та покращення естетичного вигляду додатку активно використовується модуль «styles». Цей модуль централізує конфігурацію елементів зовнішнього вигляду та інтерфейсу користувача. Спочатку функція setup_custom_styles() використовує об’єкт ttk.Style для визначення глобальних стилів для стандартних віджетів ttk, зокрема рамок (TFrame), кнопок (TButton) та міток (TLabel):

```
def setup_custom_styles():
    style = ttk.Style()
```

```
style.configure('Custom.TFrame',
        background='#f0f0f0')
style.configure('Custom.TButton',
        font=('Segoe UI', 10),
        background='#007bff',
        foreground='white')
style.configure('Custom.TLabel',
        font=('Segoe UI', 10),
        background='#f0f0f0',
    foreground='#333333')
```

Цей код встановлює колір фону за замовчуванням (#f0f0f0), а також кольори та шрифти для основних контейнерів та інформаційних елементів, що сприяє візуальній гармонії. Шрифт, Segoe UI або DejaVu Sans, вибирається залежно від операційної системи під час запуску програми.

По-друге, модуль «styles» містить фабричні функції create_custom_button() і create_custom_label() для створення віджетів типу tk.Button і ttk.Label, які містять попередньо застосовані кастомні стилі. Це спрощує код в «main_app», гарантуючи однаковий вигляд кнопок і міток у всьому додатку. Функція create_custom_button визначається наступним чином:

```
    def create_custom_button(parent, text, command=None, kwargs):
return tk.Button(parent,
        text=text,
        command=command,
        bg='#007bff',
        fg='white',
        activebackground='#0056b3',
        activeforeground='white',
        font=('Segoe UI', 10),
        relief='raised',
        padx=10,
```

```
        pady=5,
        kwargs)
```

Ця функція створює стандартну кнопку tk.Button з чітко визначеними параметрами вигляду: синій фон (bg='#007bff'), білий текст (fg='white'), темне тло при активації (activebackground='#0056b3'), рельєфний вигляд (relief='raised') та внутрішні відступи (padx, pady) для покращення читабельності.

Додатково, для забезпечення кращої інтерактивності, у «main_app» визначено клас CustomButton, що успадковується від tk.Button. Він не лише реалізує основні стилі у конструкторі, але й включає обробники подій <Enter> та <Leave>, які змінюють колір фону кнопки при наведенні миші, надаючи користувачеві візуальний зворотній зв'язок:

```
class CustomButton(tk.Button):
    def __init__(self, master=None, kwargs):
        super().__init__(master, kwargs)
        self.configure(
            bg='#007bff',
            fg='white',
            activebackground='#0056b3',
            activeforeground='white',
            font=DEFAULT_FONT,
            relief='raised',
            padx=10,
            pady=5
        )
        self.bind('<Enter>', self.on_enter)
        self.bind('<Leave>', self.on_leave)
    def on_enter(self, e):
        self['background'] = '#0056b3'
    def on_leave(self, e):
```

self['background'] = '#007bff'

Подвійний клас робить кнопки більш динамічними та візуально привабливими.

Взаємодія користувача з системою полегшується за допомогою обробників подій, пов'язаних з елементами керування. Наприклад, натискання кнопки «Увійти» у вікні автентифікації запускає функцію, яка отримує дані з полів введення і використовує модуль «database_connection» для перевірки облікових даних (за допомогою password_utils.verify_password). Якщо перевірка пройшла успішно, відкривається головне робоче вікно. При натисканні кнопки «Обчислити» відбувається збір даних з форм введення для порівняння та виклик методів з класу AHPCalculator для виконання обчислень. Для відображення інформаційних повідомлень (наприклад, про успішну реєстрацію, помилки при вході або нагадування про перевірку узгодженості) або для отримання простого тексту від користувачів використовуються стандартні діалогові вікна з модуля tkinter, а саме messagebox і tkinter.simpledialog, які імпортуються в «main_app».

Важливим компонентом інтерфейсу є візуалізація результатів аналізу (рис. 3.6). Для цього інтегровано бібліотеку Matplotlib. У модулі «main_app» з matplotlib.backends.backend_tkagg імпортовано клас FigureCanvasTkAgg. Цей клас дозволяє створити віджет Tkinter, який відображає графіки, згенеровані за допомогою Matplotlib, такі як гістограми, що ілюструють ваги критеріїв або остаточні антивірусні пріоритети. Цей віджет вбудовується у відповідний фрейм програми для візуального представлення результатів поряд з табличними даними.

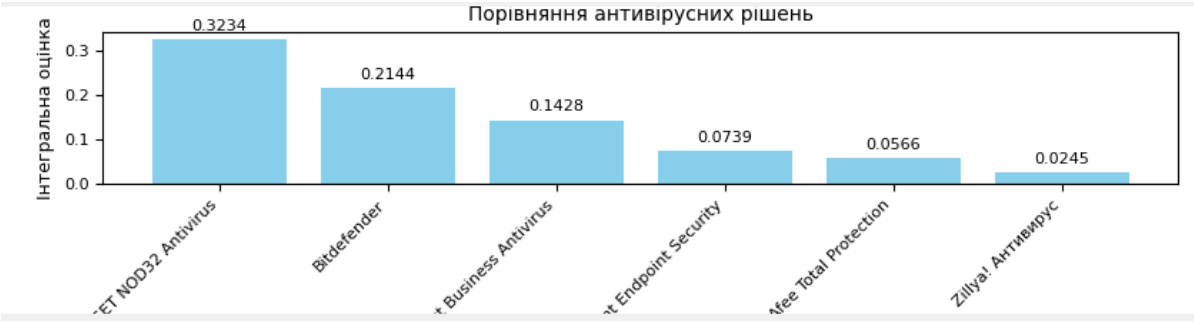

Рисунок 3.6 – Приклад графіку Matplotlib

Інтерфейс користувача в «main_app» тісно пов’язаний з іншими модулями системи, що забезпечує простий і функціональний доступ до функцій експертної системи для аналізу та вибору антивірусного програмного забезпечення.

3.4 Тестування та валідація експертної системи

Оцінка програмного забезпечення експертної системи перевіряє правильність реалізації всіх запланованих функцій і підтверджує, що система працює стабільно.

3.4.1. Тестування функціональності програмного забезпечення

При тестуванні системи використовувалися методи модульного та інтеграційного тестування.

Тестування модулів було зосереджено на конкретних компонентах системи: модулі підключення до бази даних, модулі обчислення AHP та сегментах коду, що керують елементами графічного інтерфейсу. Тести перевіряли точність функцій автентифікації користувачів, операцій з базою даних (створення, читання, оновлення та видалення записів), а також арифметичних обчислень, що лежать в основі методу AHP.

Інтеграційне тестування підтвердило, що взаємодія між різними модулями системи є точною. Це включає в себе те, як введені користувачем дані з графічного інтерфейсу передаються до модуля розрахунку AHP, як результати зберігаються в базі даних за допомогою модуля database_connection, і як ця інформація згодом витягується і відображається в інтерфейсі користувача.

Початковий крок передбачає перевірку автентифікації. Для цього ми спробуємо увійти в систему, використовуючи дійсні ім’я користувача та пароль (рис. 3.7).

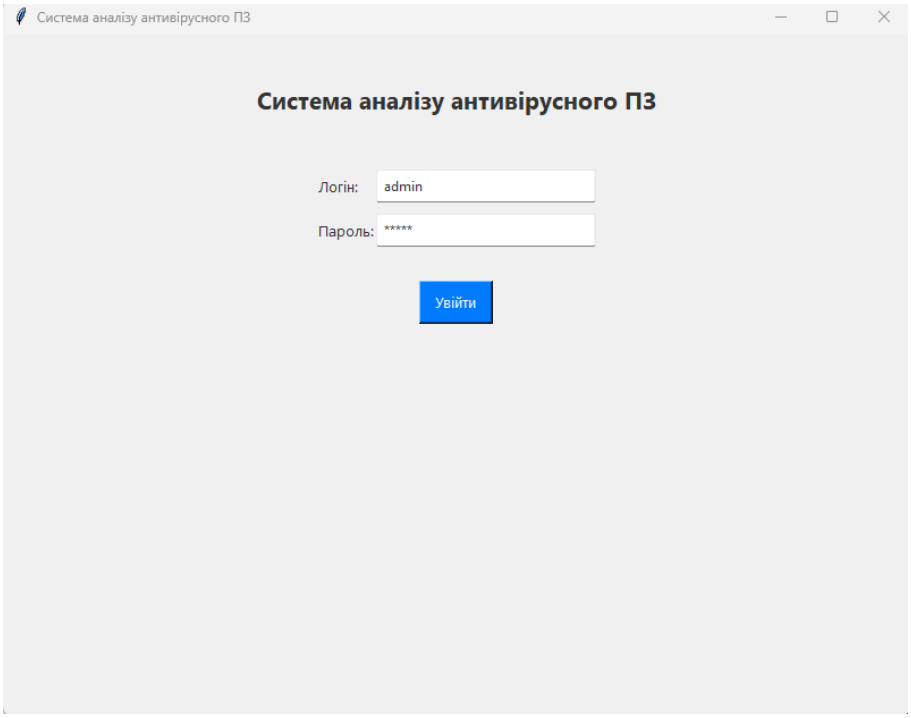

Рисунок 3.7 – Аутентифікація користувача

Якщо облікові дані введено правильно, програма відобразить головне меню де одразу можна переглянути усіх користувачів, додати нових, або ж зминити їх пароль (рис. 3.8).

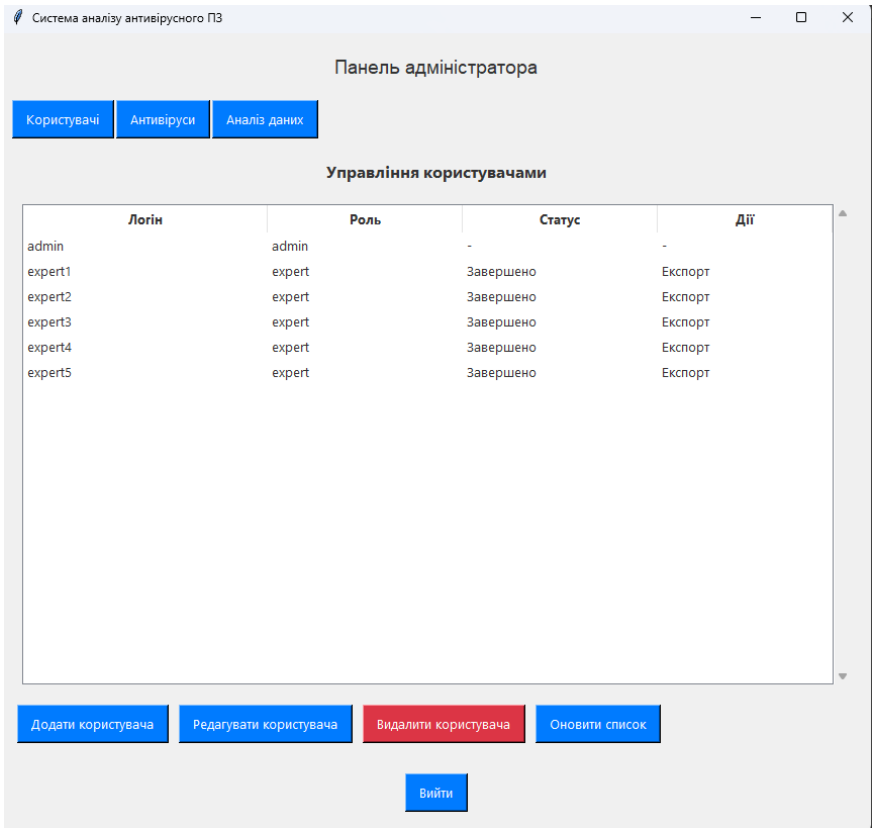

Рисунок 3.8 – Головне меню програми

Крім того, була зроблена спроба увійти в систему, використовуючи неправильні ім’я користувача та пароль. У таких випадках програма видасть повідомлення про помилку. Однак у цьому повідомленні про помилку не буде вказано, які саме дані користувач ввів неправильно (рис. 3.9).

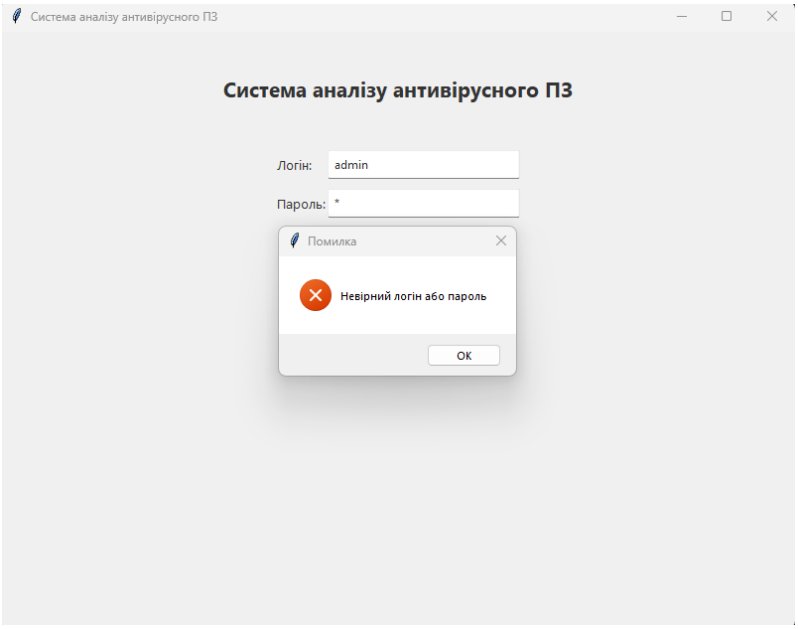

Рисунок 3.9 – Помилка при не вірному логіні та паролі

Наступним кроком буде тестування CRUD-операцій над базою даних. Для цього було додано нового користувача (рис. 3.10).

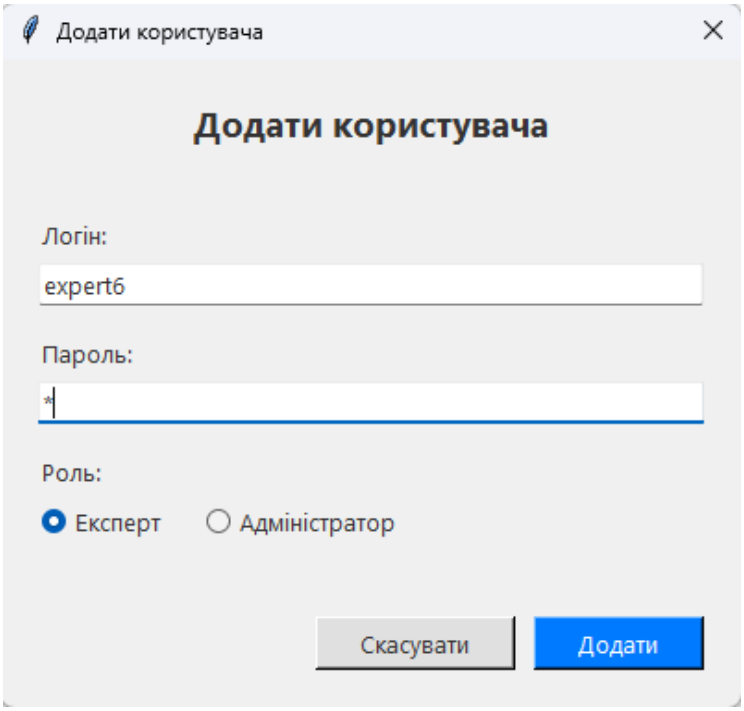

Рисунок 3.10 – Форма додавання нового користувача

Після успішної операції з’явиться повідомлення з підтвердженням, що користувача додано (рис. 3.11).

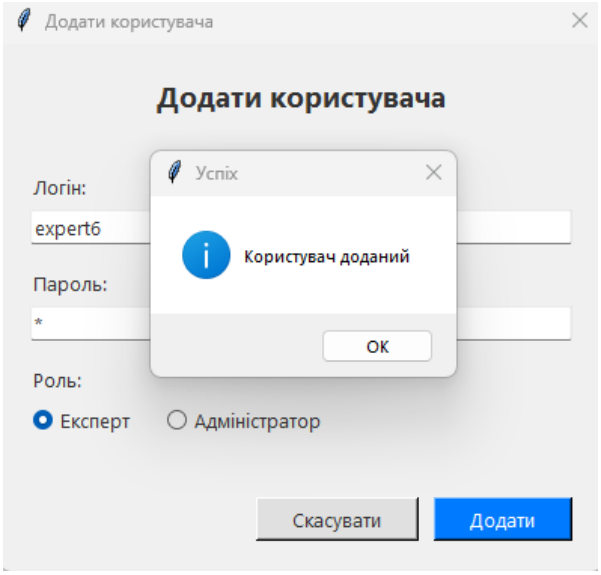

Рисунок 3.11 – Успішне додавання нового користувача

Після цього в програмі з’явиться новий користувач, в даному випадку «expert6» (рис. 3.12).

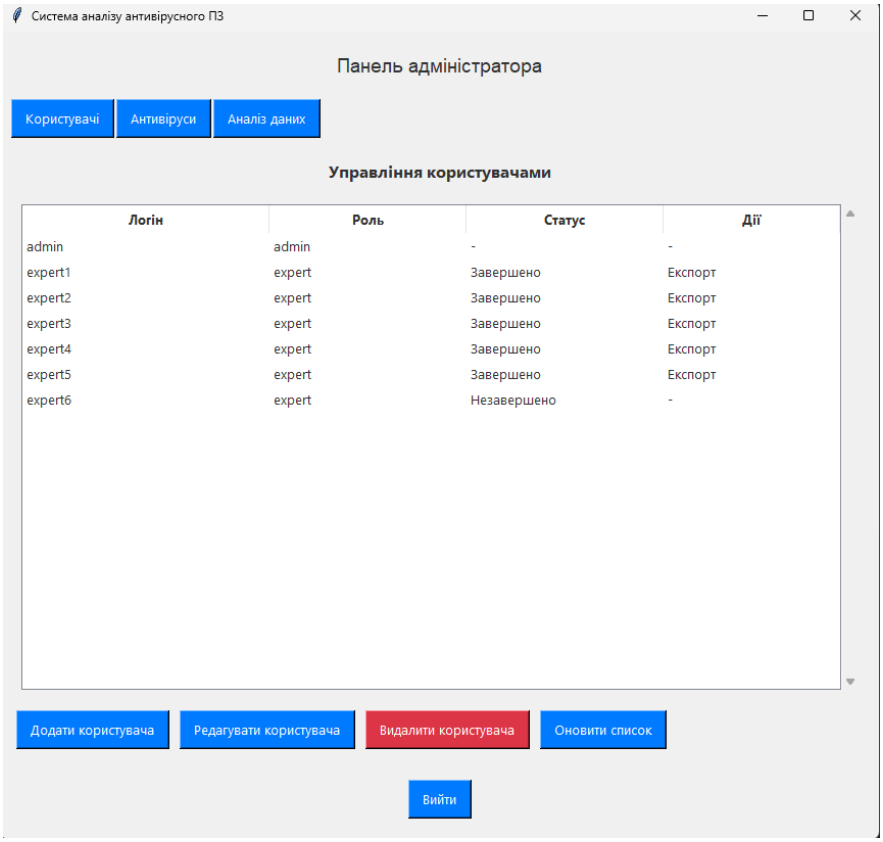

Рисунок 3.12 – Відображення нового користувача

Потім цей запис було відредаговано, щоб протестувати функцію оновлення даних користувача (рис. 3.13, 3.14).

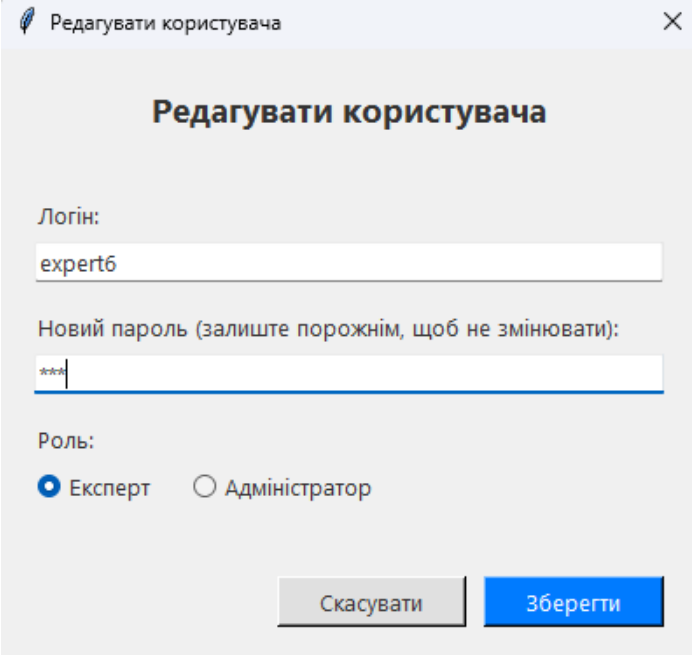

Рисунок 3.13 – Форма редагування користувача

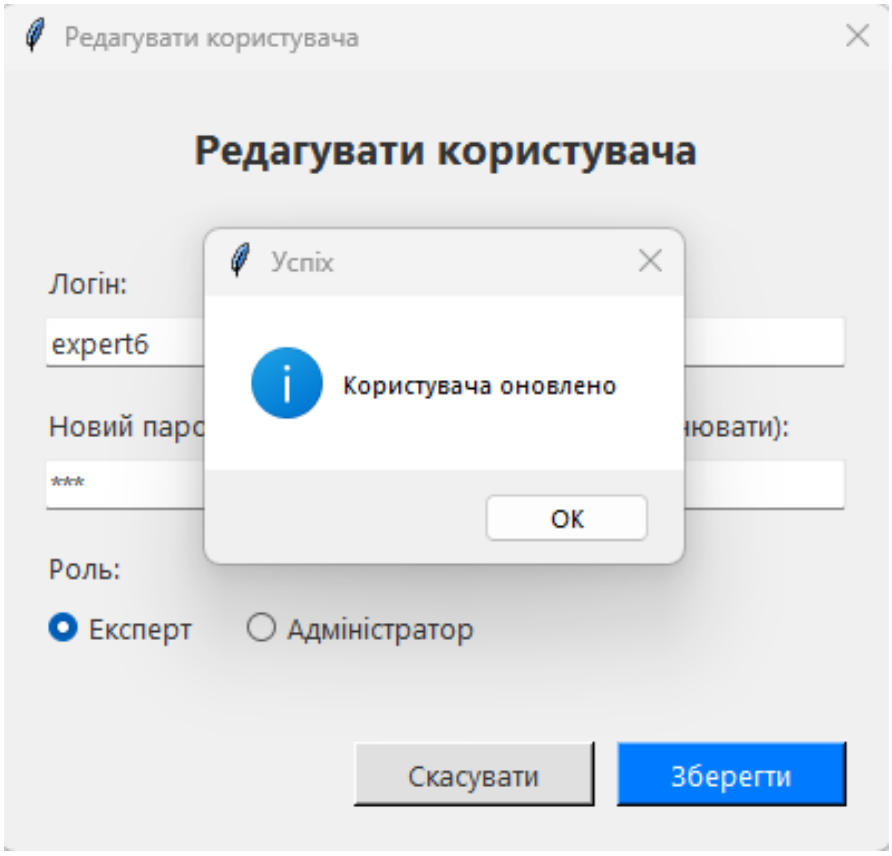

Рисунок 3.14 – Успішне редагування користувача

Далі було протестовано функцію видалення користувача, видаливши новоствореного користувача (рис. 3.15). Під час цього процесу з’являється вікно (див. рис. 3.9).

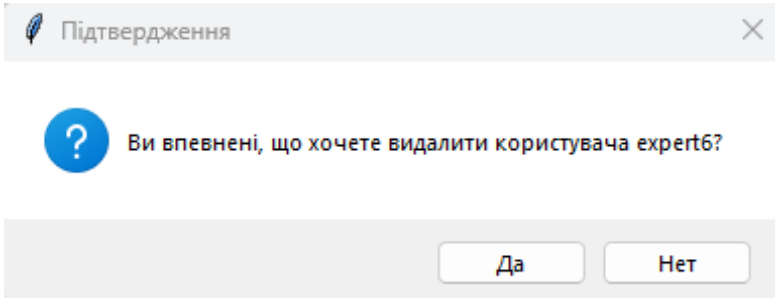

Рисунок 3.15 – Видалення користувача

Після видалення користувач зникає з головного меню програми (рис. 3.16).

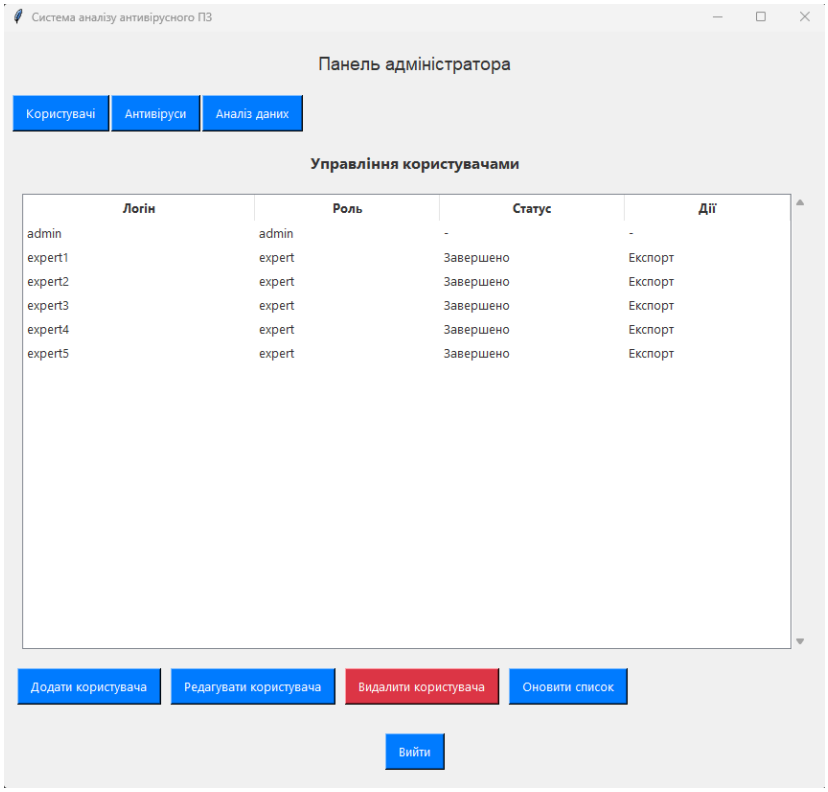

Рисунок 3.16 – Головне меню після видалення користувача

Тестування підтвердило ефективність основних функціональних компонентів програмного забезпечення та їхню взаємодію. Аналіз результатів за ключовими функціями, згаданими вище, показав, що система працює коректно, обробляє аутентифікацію, дозволяє виконувати основні операції з даними в базі даних і пропонує зручний графічний інтерфейс для взаємодії з користувачем.

Таким чином, функціональне програмне забезпечення відповідає визначеним вимогам.

3.4.2. Експериментальна апробація системи

Основною метою експериментального тестування системи було підтвердження її ефективності в оцінюванні антивірусного програмного забезпечення для критично важливих інформаційних систем з використанням експертних оцінок та врахуванням коефіцієнту ризику і значення CVSS. Експеримент мав на меті проілюструвати процес збору рейтингів за допомогою розробленого програмного забезпечення, що завершився отриманням остаточних антивірусних рейтингів, отриманих за обраною методологією.

Об’єкти аналізу, експертна група: Було досліджено шість антивірусних рішень, обраних на основі критеріїв, визначених у Переліку ДССЗЗІ. У дослідженні брала участь команда експертів у складі п’яти фахівців з відповідним досвідом у сфері кібербезпеки та захисту критичної інфраструктури.

– Експертні оцінки збиралися за допомогою спеціалізованого програмного інтерфейсу, який забезпечує стандартизований підхід до введення даних та мінімізує можливі помилки. Перед початком роботи кожен експерт пройшов коротке ознайомлення з системою та методологією оцінювання, щоб гарантувати однакове розуміння шкали порівнянь та важливості кожного етапу. Кожен експерт послідовно виконує попарні порівняння:

– Критеріїв оцінювання між собою.

– Альтернатив (антивірусів) під кожним з критеріїв К1-К6 окремо.

– Експертні оцінки у вигляді матриць попарних порівнянь були занесені в систему. У табл. 3.1 - 3.7 представлені дані які заповнив перший експерт через програмний інтерфейс, таблиці з даними інших експертів винесені у Додаток А (див. Додаток А).

Таблиця 3.1

Таблиця попарних порівнянь експерта 1

	К1: Ймовірність виявлення шкідливого коду	К2: Рівень хибних спрацьовувань	К3: Навантаження на систему	К4: Середній час реакції	К5: Частота оновлень баз даних	К6: Ризиковий коефіцієнт
К1: Ймовірність виявлення шкідливого коду	1	6	3	5	5	6
К2: Рівень хибних спрацьовувань	0,166666667	1	5	3	5	5
К3: Навантаження на систему	0,333333333	0,2	1	6	4	5
К4: Середній час реакції	0,2	0,333333333	0,166666667	1	3	5
К5: Частота оновлень баз даних	0,2	0,2	0,25	0,333333333	1	3
К6: Ризиковий коефіцієнт	0,166666667	0,2	0,2	0,2	0,333333333	1

Таблиця 3.2

Оцінка критерію К1 в антивірусах експерта 1

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус

продовження таблиці 3.2

ESET NOD32 Antivirus	1	5	3	6	4	3
Bitdefender	0,2	1	4	5	3	3
Avast Business Antivirus	0,333333333	0,25	1	3	4	5
Check Point Endpoint Security	0,166666667	0,2	0,333333333	1	5	4
McAfee Total Protection	0,25	0,333333333	0,25	0,2	1	5
Zillya! Антивирус	0,333333333	0,333333333	0,2	0,25	0,2	1

Таблиця 3.3

Оцінка критерію К2 в антивірусах експерта 1

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	4	3	5	2	4
Bitdefender	0,25	1	3	3	5	5
Avast Business Antivirus	0,333333333	0,333333333	1	4	4	6
Check Point Endpoint Security	0,2	0,333333333	0,25	1	5	4
McAfee Total Protection	0,5	0,2	0,25	0,2	1	5
Zillya! Антивирус	0,25	0,2	0,166666667	0,25	0,2	1

Таблиця 3.4

Оцінка критерію К3 в антивірусах експерта 1

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	3	4	4	5	7
Bitdefender	0,333333333	1	4	4	4	3
Avast Business Antivirus	0,25	0,25	1	3	5	6
Check Point Endpoint Security	0,25	0,25	0,333333333	1	5	4
McAfee Total Protection	0,2	0,25	0,2	0,2	1	3
Zillya! Антивирус	0,142857143	0,333333333	0,166666667	0,25	0,333333333	1

Таблиця 3.5

Оцінка критерію К4 в антивірусах експерта 1

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	4	4	5	3	5
Bitdefender	0,25	1	4	3	3	5
Avast Business Antivirus	0,25	0,25	1	4	5	5
Check Point Endpoint Security	0,2	0,333333333	0,25	1	2	5

продовження таблиці 3.5

McAfee Total Protection	0,333333333	0,333333333	0,2	0,5	1	5
Zillya! Антивирус	0,2	0,2	0,2	0,2	0,2	1

Таблиця 3.6

Оцінка критерію К5 в антивірусах експерта 1

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	5	4	5	4	4
Bitdefender	0,2	1	4	5	5	4
Avast Business Antivirus	0,25	0,25	1	4	5	5
Check Point Endpoint Security	0,2	0,2	0,25	1	5	5
McAfee Total Protection	0,25	0,2	0,2	0,2	1	6
Zillya! Антивирус	0,25	0,25	0,2	0,2	0,166666667	1

Таблиця 3.7

Оцінка критерію К6 в антивірусах експерта 1

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	4	4	5	4	5

продовження таблиці 3.7

Bitdefender	0,25	1	5	5	4	5
Avast Business Antivirus	0,25	0,2	1	4	5	7
Check Point Endpoint Security	0,2	0,2	0,25	1	5	6
McAfee Total Protection	0,25	0,25	0,2	0,2	1	6
Zillya! Антивирус	0,2	0,2	0,142857143	0,166666667	0,166666667	1

Після збору індивідуальних оцінок система об’єднала експертні матриці попарного порівняння критеріїв та альтернатив через середнє геометричне. Для агрегування використовувалася формула:

$$a_{ij}^{agg} = (\prod_{k=1}^{n} a_{ij}^{(k)})^{\frac{1}{n}} \quad (3.1)$$

де n – кількість експертів.

На основі агрегованої матриці критеріїв були розраховані їх вагові коефіцієнти. Розраховані вагові коефіцієнти критеріїв склали:

– К1: 0.4547
– К2: 0.2416
– К3: 0.1493
– К4: 0.0818
– К5: 0.0463
– К6: 0.0264

Далі були розраховані локальні вагові коефіцієнти для кожного антивірусу за кожним критерієм К1-К6 на основі відповідних агрегованих матриць альтернатив.

Остаточний рейтинг антивірусних рішень було отримано шляхом розрахунку інтегральної оцінкиP_i для кожного антивірусу з використанням

підходів А та В. Значення CVSS для антивірусів були отримані на основі даних:

– Антивірус ESET NOD32 - 0,84;

– Bitdefender Antivirus - 0.80;

– Avast Business Antivirus - 0,65;

– Check Point Endpoint Security - 0,68;

– McAfee Total Protection - 0,72 ;

– Zillya! Антивірус - 0.78.

Підхід А: Коефіцієнт ризику (К6) включається до матриці критеріїв, а інтегральна оцінка розраховується за формулою:

$$P_i = (\sum_{j=1}^{6} w_j \cdot s_{ij}) \cdot \ CVSS_i \quad (3.2)$$

де w_j – ваговий коефіцієнт критерію j, s_{ij} – локальна оцінка альтернативи i за критерієм j, а $CVSS_i$ – значення CVSS для альтернативи i.

Результати розрахунків для підходу А:

– Антивірус ESET NOD32: 0.3454;

– Антивірус Bitdefender: 0.1965;

– Avast Business Antivirus: 0.1046;

– Check Point Endpoint Security: 0.0645;

– McAfee Total Protection: 0.0404;

– Zillya! Антивірус: 0.0243.

Підхід Б: Спочатку розраховується базова оцінка B_i за критеріями К1-К5:

$$B_i = \sum_{j=1}^{5} w_j \cdot s_{ij} \quad (3.3)$$

після чого застосовується коригувальний множник для врахування ризику (R_i – локальна оцінка за критерієм К6) з коефіцієнтом $k = 0.5$:

$$P_i = (B_i \cdot (1 - k \cdot R_i)) \cdot CVSS_i \quad (3.4)$$

Результати розрахунків за підходом В (з):$k = 0.5$

– Антивірус ESET NOD32: 0.2616;

– Bitdefender Antivirus: 0.1665;

– Avast Business Antivirus: 0.0944;

– Check Point Endpoint Security: 0.0606;

– McAfee Total Protection: 0.0385;

– Zillya! Антивірус: 0.0235.

Також в рамках методології проводиться аналіз узгодженості експертних оцінок шляхом розрахунку індексу узгодженості (ІУ) для кожної матриці. Перевірка узгодженості дозволяє виявити суттєві розбіжності в судженнях експертів. У разі перевищення допустимого порогового значення індексу узгодженості методологія передбачає додаткове обговорення для коригування оцінок.

На основі підсумкових оцінок Рі, отриманих для обох підходів, можна сформувати рейтинги антивірусних рішень.

Рейтинг для підходу А:

1. Антивірус ESET NOD32 (0.3454) ;
2. Bitdefender Antivirus (0.1965) ;
3. Avast Business Antivirus (0.1046) ;
4. Check Point Endpoint Security (0.0645) ;
5. McAfee Total Protection (0.0404) ;
6. Zillya! Антивірус (0.0243).

Далі проведемо перевірку узгодженості результатів обчислень з результатом вивелення програмою. На рисунку 3.17 показано вивід програми при тих самих значеннях оцінки експертів як і при ручному обрахунку для підходу А.

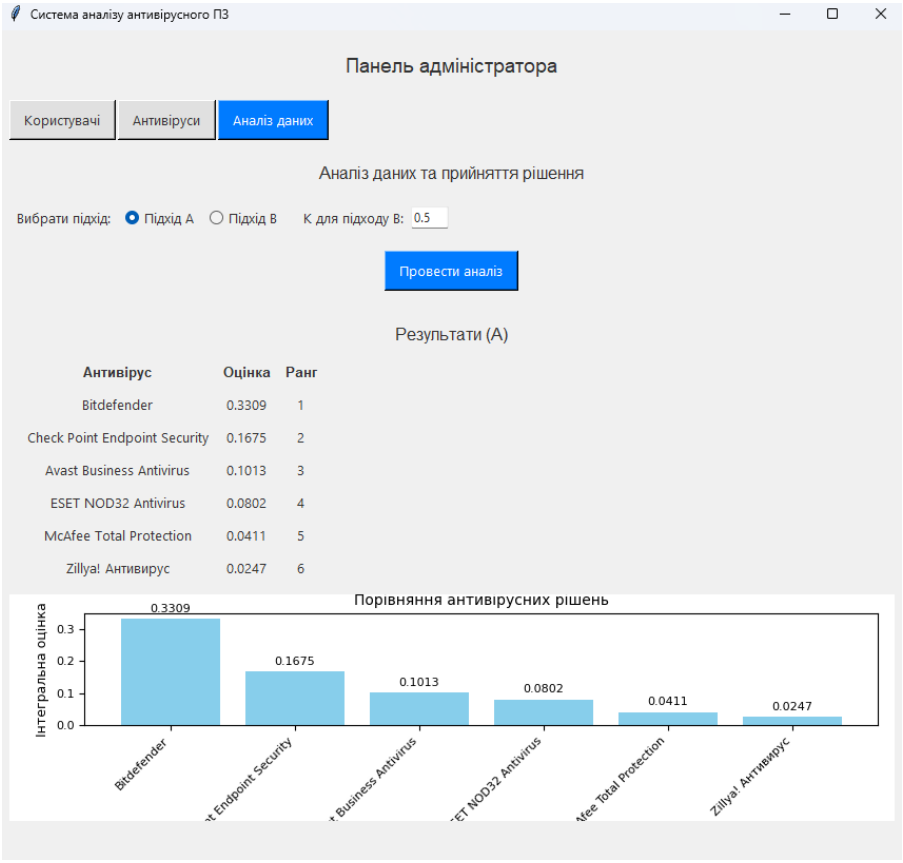

Рисунок 3.17 – результати обрахунку програми підходом А

Результати обрахунку програмаю для підходу А:

1. Антивірус ESET NOD32 (0.3309) ;
2. Bitdefender Antivirus (0.1675) ;
3. Avast Business Antivirus (0.1013) ;
4. Check Point Endpoint Security (0.0802) ;
5. McAfee Total Protection (0.0411) ;
6. Zillya! Антивірус (0.0247).

Незначні чисельні відмінності у фінальних балах порівняно з ручним обрахунком є результатом різної точності чисельних обчислень.

Рейтинг для підходу В:

1. Антивірус ESET NOD32 (0.2616) ;
2. Bitdefender Antivirus (0.1665) ;
3. Avast Business Antivirus (0.0944) ;
4. Check Point Endpoint Security (0.0606) ;

5. McAfee Total Protection (0.0385) ;

6. Zillya! Антивірус (0.0235).

Далі проведемо перевірку узгодженості результатів обчислень з результатом вивелення програмою. На рисунку 3.18 показано вивід програми при тих самих значеннях оцінки експертів як і при ручному обрахунку для підходу Б.

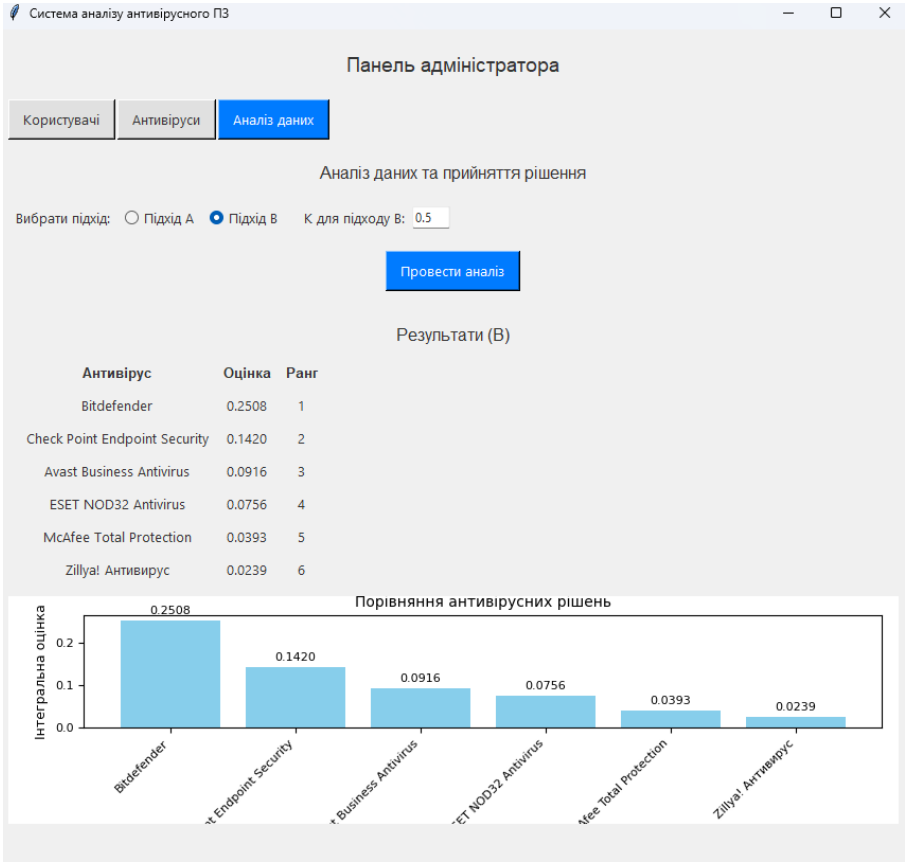

Рисунок 3.18 – Результати обрахунку програми підходом Б

Результати обрахунку програмаю для підходу Б:

1. Антивірус ESET NOD32 (0.2508) ;

2. Bitdefender Antivirus (0.1420) ;

3. Avast Business Antivirus (0.0916) ;

4. Check Point Endpoint Security (0.0756) ;

5. McAfee Total Protection (0.0393) ;

6. Zillya! Антивірус (0.0239).

Незначні чисельні відмінності у фінальних балах порівняно з ручним

обрахунком є результатом різної точності чисельних обчислень.

В обох підходах перше місце займає ESET NOD32 Antivirus, продемонструвавши високі показники за критеріями, коефіцієнтом ризику та CVSS, незалежно від застосованого методу інтеграції ризиків. Різниця в підсумкових балах між підходами пояснюється унікальним застосуванням K6, коефіцієнта ризику. Підхід А інтегрує K6 безпосередньо в загальну суму зважених критеріїв, тоді як Підхід Б використовує K6 як модифікатор до базової оцінки.

Експериментальне тестування показало, що розроблена експертна система ефективно збирає експертні оцінки через інтерфейс користувача, агрегує їх відповідно до методології AHP з урахуванням коефіцієнта ризику та CVSS двома різними методами та генерує остаточні рейтинги альтернатив. Результати узгоджуються з вхідними даними експертного порівняння (детально описані в Додатку А) та значеннями CVSS. Система ефективно виконує своє призначення - оцінювати антивірусне програмне забезпечення для критично важливих інформаційних систем.

Висновок до третього розділу

У цьому розділі описано тестування розробленого програмного забезпечення експертної системи та його експериментальну оцінку.

На етапі тестування функціональності використовувалися як модульні, так і інтеграційні методи, щоб забезпечити коректну роботу ключових компонентів системи [31]. Це включало перевірку модулів автентифікації користувачів, операцій з базами даних CRUD та функціональності графічного інтерфейсу користувача. Результати тестування підтвердили працездатність програмного забезпечення та відповідність вимогам, що стало значним практичним досягненням у процесі розробки.

Тестування системи показало її ефективність в оцінці антивірусного програмного забезпечення для критично важливих інформаційних систем. Під

час цього процесу було зібрано оцінки п’яти експертів для шести антивірусних рішень на основі шести критеріїв, включаючи коефіцієнт ризику.

Аналіз результатів тестування показав, що система ефективно обробляє експертні дані та надає впорядковану інформацію для прийняття рішення щодо вибору антивірусного програмного забезпечення, враховуючи як технічні характеристики, так і фактори ризику. Незначні числові розбіжності, виявлені між програмними та «ручними» розрахунками, пов’язані з нюансами округлення і не впливають на точність основної методології. Таким чином, експериментальна перевірка підтвердила працездатність розробленої експертної системи як інструменту багатокритеріального аналізу.

ВИСНОВКИ

В ході дослідження було ретельно проаналізовано ключові поняття, пов’язані з критичною інфраструктурою та їх класифікаціями, оцінено загрози інформаційній безпеці, що є життєво важливими для національної безпеки та економічної стабільності. Було виявлено, що класифікація активів допомагає визначити пріоритети їх захисту та розробити відповідні стратегії, а аналіз загроз підкреслює необхідність комплексного підходу до безпеки.

Розроблено методологію вибору антивірусного програмного забезпечення з використанням принципів багатокритеріального аналізу. Ця методологія включає створення математичної моделі для стандартизованого оцінювання антивірусних рішень за допомогою технічних та експертних оцінок. Ключовим компонентом є використання середнього геометричного значення для об’єднання експертних оцінок, що ефективно мінімізує суб’єктивність. Крім того, було впроваджено механізм перевірки узгодженості експертних оцінок, а також інтегровано фактор ризику, що підвищило точність і адаптивність оцінки до різних умов експлуатації. Алгоритм розробленої експертної системи пропонує структурований процес, який охоплює від збору даних до прийняття оптимального рішення.

Програмне забезпечення експертної системи пройшло ретельне функціональне та експериментальне тестування. Під час цього процесу було перевірено належну роботу життєво важливих компонентів, включаючи модулі аутентифікації, операції з базами даних та графічний інтерфейс користувача. Експериментальна оцінка продемонструвала здатність системи оцінювати антивірусне програмне забезпечення, підтвердивши, що вона ефективно обробляє експертні дані та надає впорядковану інформацію для прийняття обґрунтованих рішень на основі технічних специфікацій та факторів ризику.

Під час виконання даної роботи було розроблено детальну математичну модель для оцінки антивірусного програмного забезпечення, пристосовано до

конкретних потреб критичної інфраструктури. Вдосконалено метод багатокритеріального аналізу для вибору антивірусних рішень, додавши вагові коефіцієнти для кожного критерію. Інноваційний аспект цього дослідження полягає в удосконаленні як комплексної моделі, так і методу аналізу. Практичне значення роботи підкреслюється потенціалом розробленої експертної системи для оптимізації вибору антивірусного програмного забезпечення, тим самим підвищуючи рівень кібербезпеки об’єктів критичної інфраструктури і, можливо, зменшуючи пов’язані з цим витрати.

Подальші дослідження мають бути спрямовані на розширення бази знань експертної системи, вдосконалення алгоритмів прийняття рішень та адаптацію системи для застосування в інших сферах інформаційної безпеки, враховуючи постійну еволюцію кіберзагроз.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Герасименко, О.М. (2024) 'Критична інфраструктура України як предмет наукового пізнання: теоретичний аспект', Вісник Ужгородського національного університету. Серія: Право, 85(4), с. 42–49.

2. Про критичну інфраструктуру [Електронний ресурс]: Закон України від 16.11.2021 № 1882-IX - Режим доступу: https://zakon.rada.gov.ua/laws/show/1882-20#Text (дата звернення: 25.01.2025).

3. Галузі критичної інфраструктури України [Електронний ресурс]. - Режим доступу: https://blog.youcontrol.market/galuzi-kritichnoyi-infrastrukturi-ukrayini/ (дата звернення: 26.01.2025).

4. Порядок віднесення об'єктів до об’єктів критичної інфраструктури Вимоги до кіберзахисту об’єктів критичної інфраструктури [Електронний ресурс]. - Режим доступу: https://niss.gov.ua/sites/default/files/2019-10/fayl5-prezentaciya-ciplinskogo-peretvoreno.pdf (дата звернення: 27.01.2025).

5. Які підприємства належать до об’єктів критичної інфраструктури [Електронний ресурс]. - Режим доступу: https://profpressa.com/news/iaki-pidpriiemstva-nalezhat-do-obiektiv-kritichnoyi-infrastrukturi (дата звернення: 28.01.2025).

6. Включення підприємств, установ, організацій до переліку об’єктів критичної інфраструктури [Електронний ресурс]. - Режим доступу: https://moz.gov.ua/uploads/ckeditor/.pdf (дата звернення: 29.01.2025).

7. Деякі питання об'єктів критичної інфраструктури [Електронний ресурс]: Постанова Каб. Міністрів України від 09.10.2020 № 1109 - Режим доступу: https://zakon.rada.gov.ua/laws/show/1109-2020-п#Text (дата звернення: 30.01.2025).

8. Daniel, S.A. and Victor, S.S. (2024) 'Emerging trends in cybersecurity for critical infrastructure protection: a comprehensive review', Computer science & it research journal, 5(3), pp. 576-593.

9. Mamaghani, F. (2002) 'Evaluation and selection of an antivirus and content filtering software', Information management & computer security, 10(1), pp. 28-32.

10. Розвинена стала загроза [Електронний ресурс]. - Режим доступу: https://uk.wikipedia.org/wiki (дата звернення: 05.02.2025).

11. Удосконалення методології ранжування об’єктів критичної інфраструктури та їх віднесення до критичної інфраструктури [Електронний ресурс]. - Режим доступу: https://niss.gov.ua/sites/default/files/2016-06/krutuchna_infra-a7636.pdf (дата звернення: 01.02.2025).

12. Загрози об’єктам критичної інфраструктури України в умовах воєнного стану [Електронний ресурс]. - Режим доступу: https://visnyk-juris-uzhnu.com/wp-content/uploads/2024/09/41-2.pdf (дата звернення: 12.02.2025).

13. Про Стратегію кібербезпеки України [Електронний ресурс]: Указ Президента України від 26.08.2021 №447/2021 - Режим доступу: https://zakon.rada.gov.ua/laws/show/447/2021#Text (дата звернення: 02.02.2025).

14. Risk assessment methodologies for Critical Infrastructure Protection. Part I: A state of the art [Електронний ресурс]. - Режим доступу: https://publications.jrc.ec.europa.eu/repository/handle/JRC70046 (дата звернення: 18.02.2025).

15. Про основні засади забезпечення кібербезпеки України [Електронний ресурс]: Закон України від 05.10.2017 № 2163-VIII - Режим доступу: https://zakon.rada.gov.ua/laws/show/2163-19#Text (дата звернення: 03.02.2025).

16. ДСТУ ISO/IEC 27001:2023 [Електронний ресурс]. - Режим доступу: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=104398 (дата звернення: 05.03.2025).

17. ДСТУ ISO/IEC 27002:2023 [Електронний ресурс]. - Режим доступу: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=104399 (дата звернення: 09.03.2025).

18. Стратегія кібербезпеки України (2021 – 2025 роки) [Електронний ресурс]. - Режим доступу:

https://www.rnbo.gov.ua/files/2021/STRATEGIYA%20KYBERBEZPEKI/proekt%20strategii_kyberbezpeki_Ukr.pdf (дата звернення: 21.01.2025).

19. Методичні рекомендації щодо реагування суб’єктами забезпечення кібербезпеки на різні види подій у кіберпросторі [Електронний ресурс]. - Режим доступу: https://cip.gov.ua/ua/news/derzhspeczv-yazku-zatverdila-metodichni-rekomendaciyi-shodo-reaguvannya-sub-yektami-zabezpechennya-kiberbezpeki-na-rizni-vidi-podii-u-kiberprostori (дата звернення: 28.02.2025).

20. Herzog, C., Tong, V.V.T., Wilke, P., Van Straaten, A. and Lanet, J.L. (2020) Evasive Windows Malware: Impact on Antiviruses and Possible Countermeasures. arXiv:2009.12204.

21. Yigit, Y., Ferrag, M.A., Ghanem, M.C., Sarker, I.H., Maglaras, L.A., Chrysoulas, C., ... & Janicke, H. (2025) 'Generative AI and LLMs for critical infrastructure protection: evaluation benchmarks, agentic AI, challenges, and opportunities', Sensors, 25(6), p. 1666.

22. Mukhoid, O.V., Kostornoi, O.S. and Shyfrin, D.M. (2018) 'Selection of the Optimal Software for Designing Expert Systems', Journal of engineering sciences, (5, Iss. 2), pp. E10-E15.

23. Lu, M.T. and Guimaraes, T. (1989) 'A guide to selecting expert systems applications', Information System Management, 6(2), pp. 8-15.

24. Krisper, M., Dobaj, J. and Macher, G. (2020) 'Assessing risk estimations for cyber-security using expert judgment'. In: European Conference on Software Process Improvement. Cham: Springer International Publishing, pp. 120-134.

25. Bhol, S.G. (2025) 'Applications of Multi Criteria Decision Making Methods in Cyber Security'. In: Cyber-Physical Systems Security, pp. 233-258.

26. Moreira, F.R., Canedo, E.D., Nunes, R.R., Serrano, A.L.M., Abbas, C.J.B., Júnior, M.L.P. and de Mendonça, F.L.L. (рік не вказано) Cybersecurity Risk Assessment Through Analytic Hierarchy Process: Integrating Multicriteria and Sensitivity Analysis.

27. Котляров, О.Ю. та Бортнік, Л.Л. (2024) 'Порівняльний аналіз сучасних систем захисту віртуальних мереж та їх методології', Сучасний захист

інформації, (4), сс. 60-72.

28. Лаптєв, О.А., Гришанович, Т.О., Жолоб, Я.В. та Жигаревич, О.К. (2022) Методичні вказівки до лабораторних робіт з дисципліни «Програмно-апаратне забезпечення та захист мобільних пристроїв».

29. Bays, L.R., Oliveira, R.R., Barcellos, M.P., Gaspary, L.P. and Mauro Madeira, E.R. (2015) 'Virtual network security: threats, countermeasures, and challenges', Journal of Internet Services and Applications, 6, pp. 1-19.

30. Бучик, С.С., Кондратенко, С.О. та Писарчук, О.О. (2006) Системи підтримки прийняття рішень: конспект лекцій. Житомир: ЖВІРЕ, с. 168.

31. Ryu, Y., Shin, K., Lee, J., Jung, D.J. and Cho, H.M. (2024, January) 'Real-World Antivirus Evaluation Methodology: Applying Modern Criteria for Assessing Antivirus Functionality'. In: 2024 International Conference on Information Networking (ICOIN). IEEE, pp. 783-788.

ДОДАТОК А

Таблиця 1

Таблиця попарних порівнянь експерта 2

	К1: Ймовірність виявлення шкідливого коду	К2: Рівень хибних спрацьовувань	К3: Навантаження на систему	К4: Середній час реакції	К5: Частота оновлень баз даних	К6: Ризиковий коефіцієнт
К1: Ймовірність виявлення шкідливого коду	1	9	6	5	6	7
К2: Рівень хибних спрацьовувань	0,111111111	1	5	5	6	6
К3: Навантаження на систему	0,166666667	0,2	1	6	6	5
К4: Середній час реакції	0,2	0,2	0,166666667	1	5	5
К5: Частота оновлень баз даних	0,166666667	0,166666667	0,166666667	0,2	1	6
К6: Ризиковий коефіцієнт	0,142857143	0,166666667	0,2	0,2	0,16666667	1

Таблиця 2

Оцінка критерію K1 в антивірусах експерта 2

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	3	4	3	4	3
Bitdefender	0,333333333	1	2	4	3	4
Avast Business Antivirus	0,25	0,5	1	2	3	4
Check Point Endpoint Security	0,333333333	0,25	0,5	1	4	4
McAfee Total Protection	0,25	0,333333333	0,333333333	0,25	1	5
Zillya! Антивирус	0,333333333	0,25	0,25	0,25	0,2	1

Таблиця 3

Оцінка критерію K2 в антивірусах експерта 2

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	7	8	6	7	8
Bitdefender	0,142857143	1	5	3	7	8
Avast Business Antivirus	0,125	0,2	1	6	5	7

продовження таблиці 3

Check Point Endpoint Security	0,166666667	0,333333333	0,166666667	1	5	7
McAfee Total Protection	0,142857143	0,142857143	0,2	0,2	1	7
Zillya! Антивирус	0,125	0,125	0,142857143	0,142857143	0,142857143	1

Таблиця 4

Оцінка критерію К3 в антивірусах експерта 2

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	5	5	7	6	4
Bitdefender	0,2	1	5	5	6	6
Avast Business Antivirus	0,2	0,2	1	6	6	6
Check Point Endpoint Security	0,142857143	0,2	0,166666667	1	6	6
McAfee Total Protection	0,166666667	0,166666667	0,166666667	0,166666667	1	5
Zillya! Антивирус	0,25	0,166666667	0,166666667	0,166666667	0,2	1

Таблиця 5

Оцінка критерію К4 в антивірусах експерта 2

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	7	6	5	5	5
Bitdefender	0,142857143	1	4	5	5	5
Avast Business Antivirus	0,166666667	0,25	1	5	5	6
Check Point Endpoint Security	0,2	0,2	0,2	1	5	5
McAfee Total Protection	0,2	0,2	0,2	0,2	1	5
Zillya! Антивирус	0,2	0,2	0,166666667	0,2	0,2	1

Таблиця 6

Оцінка критерію К5 в антивірусах експерта 2

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	4	6	2	6	8
Bitdefender	0,25	1	4	5	6	6
Avast Business Antivirus	0,166666667	0,25	1	5	6	5

продовження таблиці 6

Check Point Endpoint Security	0,5	0,2	0,2	1	4	4
McAfee Total Protection	0,166666667	0,166666667	0,166666667	0,25	1	5
Zillya! Антивирус	0,125	0,166666667	0,2	0,25	0,2	1

Таблиця 7

Оцінка критерію К6 в антивірусах експерта 2

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	6	6	7	5	8
Bitdefender	0,166666667	1	6	6	7	6
Avast Business Antivirus	0,166666667	0,166666667	1	5	7	5
Check Point Endpoint Security	0,142857143	0,166666667	0,2	1	5	4
McAfee Total Protection	0,2	0,142857143	0,142857143	0,2	1	5
Zillya! Антивирус	0,125	0,166666667	0,2	0,25	0,2	1

Таблиця 8

Таблиця попарних порівнянь експерта 3

	К1: Ймовірність виявлення шкідливого коду	К2: Рівень хибних спрацьовувань	К3: Навантаження на систему	К4: Середній час реакції	К5: Частота оновлень баз даних	К6: Ризиковий коефіцієнт
К1: Ймовірність виявлення шкідливого коду	1	7	6	5	4	7
К2: Рівень хибних спрацьовувань	0,142857143	1	6	6	6	6
К3: Навантаження на систему	0,166666667	0,166666667	1	6	7	6
К4: Середній час реакції	0,2	0,166666667	0,166666667	1	7	4
К5: Частота оновлень баз даних	0,25	0,166666667	0,142857143	0,142857143	1	7
К6: Ризиковий коефіцієнт	0,142857143	0,166666667	0,166666667	0,25	0,1428571 4	1

Таблиця 9

Оцінка критерію К1 в антивірусах експерта 3

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	5	2	4	4	4
Bitdefender	0,2	1	4	4	3	5

продовження таблиці 9

Avast Business Antivirus	0,5	0,25	1	4	2	6
Check Point Endpoint Security	0,25	0,25	0,25	1	4	5
McAfee Total Protection	0,25	0,333333333	0,5	0,25	1	5
Zillya! Антивирус	0,25	0,2	0,166666667	0,2	0,2	1

Таблиця 10

Оцінка критерію К2 в антивірусах експерта 3

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	7	5	5	5	2
Bitdefender	0,142857143	1	5	6	6	5
Avast Business Antivirus	0,2	0,2	1	5	5	5
Check Point Endpoint Security	0,2	0,166666667	0,2	1	6	4
McAfee Total Protection	0,2	0,166666667	0,2	0,166666667	1	5
Zillya! Антивирус	0,5	0,2	0,2	0,25	0,2	1

Таблиця 11

Оцінка критерію К3 в антивірусах експерта 3

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	5	4	4	3	6
Bitdefender	0,2	1	5	5	6	3
Avast Business Antivirus	0,25	0,2	1	5	4	5
Check Point Endpoint Security	0,25	0,2	0,2	1	4	5
McAfee Total Protection	0,333333333	0,166666667	0,25	0,25	1	5
Zillya! Антивирус	0,166666667	0,333333333	0,2	0,2	0,2	1

Таблиця 12

Оцінка критерію К4 в антивірусах експерта 3

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	7	5	6	6	6
Bitdefender	0,142857143	1	7	6	6	7
Avast Business Antivirus	0,2	0,142857143	1	6	6	8

продовження таблиці 12

Check Point Endpoint Security	0,166666667	0,166666667	0,166666667	1	6	7
McAfee Total Protection	0,166666667	0,166666667	0,166666667	0,166666667	1	8
Zillya! Антивирус	0,166666667	0,142857143	0,125	0,142857143	0,125	1

Таблиця 13

Оцінка критерію K5 в антивірусах експерта 3

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	6	4	7	6	8
Bitdefender	0,166666667	1	6	6	6	8
Avast Business Antivirus	0,25	0,166666667	1	6	6	7
Check Point Endpoint Security	0,142857143	0,166666667	0,166666667	1	5	3
McAfee Total Protection	0,166666667	0,166666667	0,166666667	0,2	1	6
Zillya! Антивирус	0,125	0,125	0,142857143	0,333333333	0,166666667	1

Таблиця 14

Оцінка критерію K6 в антивірусах експерта 3

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	4	4	7	6	8

продовження таблиці 14

Bitdefender	0,25	1	6	5	6	7
Avast Business Antivirus	0,25	0,166666667	1	7	5	7
Check Point Endpoint Security	0,142857143	0,2	0,142857143	1	5	4
McAfee Total Protection	0,166666667	0,166666667	0,2	0,2	1	5
Zillya! Антивирус	0,125	0,142857143	0,142857143	0,25	0,2	1

Таблиця 15

Таблиця попарних порівнянь експерта 4

	К1: Ймовірність виявлення шкідливого коду	К2: Рівень хибних спрацьовувань	К3: Навантаження на систему	К4: Середній час реакції	К5: Частота оновлень баз даних	К6: Ризиковий коефіцієнт
К1: Ймовірність виявлення шкідливого коду	1	6	4	6	5	7
К2: Рівень хибних спрацьовувань	0,166666667	1	6	7	5	7
К3: Навантаження на систему	0,25	0,166666667	1	5	6	3
К4: Середній час реакції	0,166666667	0,142857143	0,2	1	7	5

продовження таблиці 15

К5: Частота оновлень баз даних	0,2	0,2	0,166666667	0,142857143	1	5
К6: Ризиковий коефіцієнт	0,142857143	0,142857143	0,333333333	0,2	0,2	1

Таблиця 16

Оцінка критерію К1 в антивірусах експерта 4

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	5	2	4	6	5
Bitdefender	0,2	1	5	5	5	5
Avast Business Antivirus	0,5	0,2	1	5	6	6
Check Point Endpoint Security	0,25	0,2	0,2	1	6	4
McAfee Total Protection	0,166666667	0,2	0,166666667	0,166666667	1	6
Zillya! Антивирус	0,2	0,2	0,166666667	0,25	0,166666667	1

Таблиця 17

Оцінка критерію К2 в антивірусах експерта 4

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	5	5	6	5	6

продовження таблиці 17

Avast Business Antivirus	0,2	0,2	1	6	6	6
Check Point Endpoint Security	0,166666667	0,166666667	0,166666667	1	4	6
McAfee Total Protection	0,2	0,2	0,166666667	0,25	1	6
Zillya! Антивирус	0,166666667	0,333333333	0,166666667	0,166666667	0,166666667	1

Таблиця 18

Оцінка критерію К3 в антивірусах експерта 4

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	5	5	5	5	6
Bitdefender	0,2	1	6	5	4	5
Avast Business Antivirus	0,2	0,166666667	1	5	5	3
Check Point Endpoint Security	0,2	0,2	0,2	1	5	7
McAfee Total Protection	0,2	0,25	0,2	0,2	1	6
Zillya! Антивирус	0,166666667	0,2	0,333333333	0,142857143	0,166666667	1

Таблиця 19

Оцінка критерію К4 в антивірусах експерта 4

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	6	5	5	4	5
Bitdefender	0,166666667	1	5	5	6	4
Avast Business Antivirus	0,2	0,2	1	5	4	5
Check Point Endpoint Security	0,2	0,2	0,2	1	5	5
McAfee Total Protection	0,25	0,166666667	0,25	0,2	1	4
Zillya! Антивирус	0,2	0,25	0,2	0,2	0,25	1

Таблиця 20

Оцінка критерію К5 в антивірусах експерта 4

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	6	6	6	5	4
Bitdefender	0,166666667	1	6	6	6	5

продовження таблиці 20

Avast Business Antivirus	0,166666667	0,166666667	1	7	6	6
Check Point Endpoint Security	0,166666667	0,166666667	0,142857143	1	6	6
McAfee Total Protection	0,2	0,166666667	0,166666667	0,166666667	1	7
Zillya! Антивирус	0,25	0,2	0,166666667	0,166666667	0,142857143	1

Таблиця 21

Оцінка критерію К6 в антивірусах експерта 4

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	6	6	5	5	5
Bitdefender	0,166666667	1	5	5	5	7
Avast Business Antivirus	0,166666667	0,2	1	5	6	7
Check Point Endpoint Security	0,2	0,2	0,2	1	4	5
McAfee Total Protection	0,2	0,2	0,166666667	0,25	1	6
Zillya! Антивирус	0,2	0,142857143	0,142857143	0,2	0,166666667	1

Таблиця 22

Таблиця попарних порівнянь експерта 5

	К1: Ймовірність виявлення шкідливого коду	К2: Рівень хибних спрацьовувань	К3: Навантаження на систему	К4: Середній час реакції	К5: Частота оновлень баз даних	К6: Ризиковий коефіцієнт
К1: Ймовірність виявлення шкідливого коду	1	6	4	5	7	6
К2: Рівень хибних спрацьовувань	0,166666667	1	4	5	4	7
К3: Навантаження на систему	0,25	0,25	1	6	6	5
К4: Середній час реакції	0,2	0,2	0,166666667	1	7	6
К5: Частота оновлень баз даних	0,142857143	0,25	0,166666667	0,142857143	1	6
К6: Ризиковий коефіцієнт	0,166666667	0,142857143	0,2	0,166666667	0,166666667	1

Таблиця 23

Оцінка критерію К1 в антивірусах експерта 5

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус

продовження таблиці 23

ESET NOD32 Antivirus	1	6	5	4	5	5
Bitdefender	0,166666667	1	5	4	6	5
Avast Business Antivirus	0,2	0,2	1	5	8	5
Check Point Endpoint Security	0,25	0,25	0,2	1	6	7
McAfee Total Protection	0,2	0,166666667	0,125	0,166666667	1	5
Zillya! Антивирус	0,2	0,2	0,2	0,142857143	0,2	1

Таблиця 24

Оцінка критерію К2 в антивірусах експерта 5

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	5	4	5	3	5
Bitdefender	0,2	1	5	5	4	5
Avast Business Antivirus	0,25	0,2	1	4	5	5
Check Point Endpoint Security	0,2	0,2	0,25	1	5	4
McAfee Total Protection	0,333333333	0,25	0,2	0,2	1	6
Zillya! Антивирус	0,2	0,2	0,2	0,25	0,166666667	1

Таблиця 25

Оцінка критерію К3 в антивірусах експерта 5

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	6	4	5	4	6
Bitdefender	0,166666667	1	5	5	5	6
Avast Business Antivirus	0,25	0,2	1	5	7	6
Check Point Endpoint Security	0,2	0,2	0,2	1	5	6
McAfee Total Protection	0,25	0,2	0,142857143	0,2	1	5
Zillya! Антивирус	0,166666667	0,166666667	0,166666667	0,166666667	0,2	1

Таблиця 26

Оцінка критерію К4 в антивірусах експерта 5

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	6	4	6	7	7
Bitdefender	0,166666667	1	6	6	6	6
Avast Business Antivirus	0,25	0,166666667	1	5	7	7
Check Point Endpoint Security	0,166666667	0,166666667	0,2	1	6	5

продовження таблиці 26

McAfee Total Protection	0,142857143	0,166666667	0,142857143	0,166666667	1	5
Zillya! Антивирус	0,142857143	0,166666667	0,142857143	0,2	0,2	1

Таблиця 27

Оцінка критерію К5 в антивірусах експерта 5

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус
ESET NOD32 Antivirus	1	6	5	6	7	7
Bitdefender	0,166666667	1	6	5	6	7
Avast Business Antivirus	0,2	0,166666667	1	5	6	5
Check Point Endpoint Security	0,166666667	0,2	0,2	1	7	9
McAfee Total Protection	0,142857143	0,166666667	0,166666667	0,142857143	1	5
Zillya! Антивирус	0,142857143	0,142857143	0,2	0,111111111	0,2	1

Таблиця 28

Оцінка критерію К6 в антивірусах експерта 5

	ESET NOD32 Antivirus	Bitdefender	Avast Business Antivirus	Check Point Endpoint Security	McAfee Total Protection	Zillya! Антивирус

продовження таблиці 28

ESET NOD32 Antivirus	1	6	6	5	5	6
Bitdefender	0,166666667	1	6	4	6	6
Avast Business Antivirus	0,166666667	0,166666667	1	5	8	6
Check Point Endpoint Security	0,2	0,25	0,2	1	4	6
McAfee Total Protection	0,2	0,166666667	0,125	0,25	1	7
Zillya! Антивирус	0,166666667	0,166666667	0,166666667	0,166666667	0,142857143	1

ДОДАТОК Б

Код програми

Код main_app.py

```
import tkinter as tk
from tkinter import ttk, messagebox, simpledialog, filedialog
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import os
import sys
from pathlib import Path
from bson.objectid import ObjectId
import platform
import pandas as pd
from datetime import datetime
import openpyxl
current_dir = Path(os.path.dirname(os.path.abspath(__file__)))
sys.path.append(str(current_dir))
from database_connection import Database
from ahp_calculator import AHPCalculator
from styles import setup_custom_styles, create_custom_button, create_custom_label
if platform.system() == 'Linux':
    DEFAULT_FONT = ('DejaVu Sans', 10)
    BUTTON_BG = '#007bff'
    BUTTON_FG = 'white'
else:
    DEFAULT_FONT = ('Segoe UI', 10)
```

```
    BUTTON_BG = '#007bff'
    BUTTON_FG = 'white'
class CustomButton(tk.Button):
    def __init__(self, master=None, **kwargs):
        super().__init__(master, **kwargs)
        self.configure(
            bg='#007bff',
            fg='white',
            activebackground='#0056b3',
            activeforeground='white',
            font=DEFAULT_FONT,
            relief='raised',
            padx=10,
            pady=5
        )
        self.bind('<Enter>', self.on_enter)
        self.bind('<Leave>', self.on_leave)
    def on_enter(self, e):
        self['background'] = '#0056b3'
    def on_leave(self, e):
        self['background'] = '#007bff'
class CustomNotebook(ttk.Notebook):
    def __init__(self, master=None, **kwargs):
        super().__init__(master, **kwargs)
        self.configure(style="Custom.TNotebook")
class CustomTabFrame(ttk.Frame):
    def __init__(self, master=None, **kwargs):
        super().__init__(master, **kwargs)
        self.tabs = {}
        self.current_tab = None
```

```
        self.tab_buttons = {}
        self.button_frame = ttk.Frame(self)
        self.button_frame.pack(fill=tk.X, pady=(0, 10))
        self.content_frame = ttk.Frame(self)
        self.content_frame.pack(fill=tk.BOTH, expand=True)
    def add_tab(self, name, frame):
        self.tabs[name] = frame
        frame.pack_forget()
        btn = create_custom_button(self.button_frame,
                text=name,
                command=lambda: self.show_tab(name))
        btn.pack(side=tk.LEFT, padx=(0, 2))
        self.tab_buttons[name] = btn
        if not self.current_tab:
            self.show_tab(name)
    def show_tab(self, name):
        if self.current_tab:
            self.tabs[self.current_tab].pack_forget()
            self.tab_buttons[self.current_tab].configure(
                bg='#e0e0e0',
                fg='#333333'
            )
        self.tabs[name].pack(in_=self.content_frame, fill=tk.BOTH,
expand=True)
        self.tab_buttons[name].configure(
            bg='#007bff',
            fg='white'
        )
        self.current_tab = name
```

```
class AntivirusAnalysisApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Система аналізу антивірусного ПЗ")
        self.root.geometry("800x600")
        setup_custom_styles()
        screen_width = self.root.winfo_screenwidth()
        screen_height = self.root.winfo_screenheight()
        x = (screen_width - 800) // 2
        y = (screen_height - 600) // 2
        self.root.geometry(f"800x600+{x}+{y}")
        self.db = Database()
        self.ahp = AHPCalculator()
        self.current_user = None
        self.criteria_names = [
            "К1: Ймовірність виявлення шкідливого коду",
            "К2: Рівень хибних спрацьовувань",
            "К3: Навантаження на систему",
            "К4: Середній час реакції",
            "К5: Частота оновлень баз даних",
            "К6: Ризиковий коефіцієнт"]
        self.setup_login_frame()
    def setup_login_frame(self):
        self.clear_window()
        screen_width = self.root.winfo_screenwidth()
        screen_height = self.root.winfo_screenheight()
        x = (screen_width - 800) // 2
        y = (screen_height - 600) // 2
        self.root.geometry(f"800x600+{x}+{y}")
```

```
        self.login_frame = ttk.Frame(self.root, padding=20,
style="Custom.TFrame")
        self.login_frame.pack(expand=True, fill=tk.BOTH)
        create_custom_label(self.login_frame,
            text="Система аналізу антивірусного ПЗ",
            font_size=16,
            is_bold=True).pack(pady=20)
        login_form = ttk.Frame(self.login_frame, style="Custom.TFrame")
        login_form.pack(pady=20)
        create_custom_label(login_form,
            text="Логін:").grid(row=0, column=0, sticky='w', pady=5)
        self.username_var = tk.StringVar()
        ttk.Entry(login_form,
            textvariable=self.username_var,
            width=30,
            style="Custom.TEntry").grid(row=0, column=1, pady=5)
        create_custom_label(login_form,
            text="Пароль:").grid(row=1, column=0, sticky='w', pady=5)
        self.password_var = tk.StringVar()
        password_entry = ttk.Entry(login_form,
            textvariable=self.password_var,
            show="*",
            width=30,
            style="Custom.TEntry")
        password_entry.grid(row=1, column=1, pady=5)
        password_entry.bind('<Return>', lambda event: self.login())
        buttons_frame = ttk.Frame(login_form, style="Custom.TFrame")
        buttons_frame.grid(row=2, column=0, columnspan=2, pady=20)
        create_custom_button(buttons_frame,
                             text="Увійти",
```

```
                command=self.login).pack(pady=5)
    def login(self):
        username = self.username_var.get()
        password = self.password_var.get()
        if not username or not password:
            messagebox.showerror("Помилка", "Введіть логін та пароль")
            return
        user = self.db.get_user(username, password)
        if user:
            self.current_user = user
            if user['role'] == 'admin':
                self.setup_admin_interface()
            else:
                self.setup_expert_interface()
        else:
            messagebox.showerror("Помилка", "Невірний логін або пароль")
    def register(self):
        username = self.username_var.get()
        password = self.password_var.get()
        if not username or not password:
            messagebox.showerror("Помилка", "Введіть логін та пароль")
            return
        if self.db.add_user(username, password):
            messagebox.showinfo("Успіх", "Користувач зареєстрований")
        else:
            messagebox.showerror("Помилка", "Користувач вже існує")
    def clear_window(self):
        for widget in self.root.winfo_children():
            widget.destroy()
    def setup_expert_interface(self):
```

```
self.clear_window()
screen_width = self.root.winfo_screenwidth()
screen_height = self.root.winfo_screenheight()
x = (screen_width - 860) // 2
y = (screen_height - 790) // 2
self.root.geometry(f"860x790+{x}+{y}")
self.expert_frame = ttk.Frame(self.root, padding=10)
self.expert_frame.pack(fill=tk.BOTH, expand=True)
ttk.Label(self.expert_frame,
        text=f"Експерт: {self.current_user['username']}",
        font=('Segoe UI', 14, 'bold'),
        style="Title.TLabel").pack(pady=10)
tab_frame = CustomTabFrame(self.expert_frame)
tab_frame.pack(fill=tk.BOTH, expand=True, pady=10)
criteria_tab = ttk.Frame(tab_frame)
alternatives_tab = ttk.Frame(tab_frame)
tab_frame.add_tab("Порівняння критеріїв", criteria_tab)
tab_frame.add_tab("Порівняння альтернатив", alternatives_tab)
self.setup_criteria_comparison(criteria_tab)
self.setup_alternatives_comparison(alternatives_tab)
tk.Button(self.expert_frame,
        text="Вийти",
        command=self.setup_login_frame,
        bg='#007bff',
        fg='white',
        activebackground='#0056b3',
        activeforeground='white',
        font=('Segoe UI', 10),
        relief='raised',
        padx=20,
```

```
                pady=5).pack(pady=10)
        def setup_criteria_comparison(self, parent):
            ttk.Label(parent,
                text="Попарне порівняння критеріїв",
                font=('Segoe UI', 12, 'bold'),
                style="Title.TLabel").pack(pady=10)
            comparison_frame = ttk.Frame(parent)
            comparison_frame.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)
            self.criteria_matrix = self.load_or_create_matrix('criteria',
len(self.criteria_names))
            self.create_comparison_table(comparison_frame, self.criteria_names,
self.criteria_matrix, is_criteria=True)
            tk.Button(parent,
                text="Зберегти",
                command=lambda: self.save_comparison('criteria'),
                bg='#007bff',
                fg='white',
                activebackground='#0056b3',
                activeforeground='white',
                font=('Segoe UI', 10),
                relief='raised',
                padx=20,
                pady=5).pack(pady=10)
        def setup_alternatives_comparison(self, parent):
            self.alternatives = self.db.get_antiviruses()
            alt_names = [alt['name'] for alt in self.alternatives]
            criteria_selection_frame = ttk.Frame(parent)
            criteria_selection_frame.pack(fill=tk.X, pady=10)
            selection_frame = ttk.Frame(criteria_selection_frame)
            selection_frame.pack(fill=tk.X, pady=(0, 5))
```

```
            create_custom_label(selection_frame,
                    text="Вибрати критерій:",
                    font_size=10).pack(side=tk.LEFT, padx=5)
            self.selected_criterion = tk.StringVar()
            criteria_combobox = ttk.Combobox(selection_frame,
                                textvariable=self.selected_criterion,
                                values=["К1", "К2", "К3", "К4", "К5", "К6"],
                                font=('Segoe UI', 10),
                                width=10)
            criteria_combobox.pack(side=tk.LEFT, padx=5)
            criteria_combobox.current(0)
            create_custom_button(selection_frame,
                    text="Показати",
                     command=lambda:
self.show_alternatives_comparison(alt_names)).pack(side=tk.LEFT, padx=5)
            self.criterion_name_label = create_custom_label(criteria_selection_frame,
                                                text="",
                                                font_size=10,
                                                is_bold=True)
            self.criterion_name_label.pack(anchor=tk.W, padx=5, pady=(0, 5))
            def update_criterion_name(*args):
               criterion = self.selected_criterion.get()
               criterion_names = {
                  "К1": "Ймовірність виявлення шкідливого коду",
                  "К2": "Рівень хибних спрацьовувань",
                  "К3": "Навантаження на систему",
                  "К4": "Середній час реакції",
                  "К5": "Частота оновлень баз даних",
                  "К6": "Ризиковий коефіцієнт"
               }
```

```
            self.criterion_name_label.configure(text=criterion_names.get(criterion,
""))
        self.selected_criterion.trace_add("write", update_criterion_name)
        update_criterion_name()
        self.alt_comparison_frame = ttk.Frame(parent)
        self.alt_comparison_frame.pack(fill=tk.BOTH, expand=True, padx=10,
pady=10)
        self.show_alternatives_comparison(alt_names)
    def show_alternatives_comparison(self, alt_names):
        for widget in self.alt_comparison_frame.winfo_children():
            widget.destroy()
        criterion = self.selected_criterion.get()
        ttk.Label(self.alt_comparison_frame,
                  text=f"Попарне порівняння альтернатив за критерієм
{criterion}",
                  font=('Segoe UI', 12, 'bold'),
                  style="Title.TLabel").pack(pady=10)
        matrix_key = f'alt_by_{criterion}'
        self.alt_matrix = self.load_or_create_matrix(matrix_key,
len(self.alternatives))
        self.create_comparison_table(self.alt_comparison_frame, alt_names,
self.alt_matrix, is_criteria=False)
        tk.Button(self.alt_comparison_frame,
                  text="Зберегти",
                  command=lambda: self.save_comparison(matrix_key),
                  bg='#007bff',
                  fg='white',
                  activebackground='#0056b3',
                  activeforeground='white',
                  font=('Segoe UI', 10),
```

```
                relief='raised',
                padx=20,
                pady=5).pack(pady=10)
        def load_or_create_matrix(self, matrix_type, size):
            try:
                comparison = self.db.get_expert_comparison(self.current_user['username'], matrix_type)
                if comparison and 'matrix' in comparison:
                    loaded_matrix = comparison['matrix']
                    new_matrix = [[1 for _ in range(size)] for _ in range(size)]
                    for i in range(min(len(loaded_matrix), size)):
                        for j in range(min(len(loaded_matrix[i]), size)):
                            new_matrix[i][j] = loaded_matrix[i][j]
                    return new_matrix
                matrix = [[1 for _ in range(size)] for _ in range(size)]
                return matrix
            except Exception as e:
                print(f"Помилка при завантаженні/створенні матриці: {str(e)}")
                return [[1 for _ in range(size)] for _ in range(size)]
        def create_comparison_table(self, parent, labels, matrix, is_criteria=False):
            table_frame = ttk.Frame(parent, style="Custom.TFrame")
            table_frame.pack(fill=tk.BOTH, expand=True)
            explanation = ttk.Label(parent,
                                    text="Шкала порівняння: 1-однаково, 3-помірна перевага, 5-сильна перевага, " +
                                         "7-дуже сильна перевага, 9-абсолютна перевага",
                                    style="Custom.TLabel")
            explanation.pack(pady=5)
            n = len(labels)
            if len(matrix) != n:
```

```
            matrix = [[1 for _ in range(n)] for _ in range(n)]
        else:
            for i in range(n):
                if len(matrix[i]) != n:
                    matrix[i] = [1 for _ in range(n)]
        ttk.Label(table_frame, text="", style="Custom.TLabel").grid(row=0,
column=0)
        for i, label in enumerate(labels):
            ttk.Label(table_frame,
                      text=label,
                      wraplength=100,
                      style="Custom.TLabel").grid(row=0, column=i+1, padx=2,
pady=2)
        if is_criteria:
            self.criteria_entries = []
            matrix_entries = self.criteria_entries
        else:
            self.alt_entries = []
            matrix_entries = self.alt_entries
        string_vars = [[tk.StringVar() for _ in range(n)] for _ in range(n)]
        for i in range(n):
            for j in range(n):
                if i == j:
                    string_vars[i][j].set("1")
                else:
                    value = matrix[i][j]
                    if value < 1:
                        string_vars[i][j].set(f"1/{int(1/value)}")
                    else:
                        string_vars[i][j].set(str(int(value)))
```

```
            for i in range(n):
                row_entries = []
                ttk.Label(table_frame,
                          text=labels[i],
                          wraplength=100,
                          style="Custom.TLabel").grid(row=i+1,      column=0,      padx=2,
pady=2)
                for j in range(n):
                    if i == j:
                        label = ttk.Label(table_frame,
                                          text="1",
                                          style="Custom.TLabel")
                        label.grid(row=i+1, column=j+1, padx=2, pady=2)
                        row_entries.append(label)
                    elif i < j:
                        var = string_vars[i][j]
                        entry = ttk.Combobox(table_frame,
                                             values=["1", "2", "3", "4", "5", "6", "7", "8", "9"],
                                             width=5,
                                             style="Custom.TCombobox")
                        entry.grid(row=i+1, column=j+1, padx=2, pady=2)
                        current_value = var.get()
                        if current_value.startswith('1/'):
                            entry.set(current_value)
                        else:
                            entry.set(current_value)
                            if hasattr(entry, 'opposite_var'):
                                entry.opposite_var.set(f"1/{current_value}")

                        entry.row_index = i
```

```
                    entry.col_index = j
                    entry.is_criteria = is_criteria
                    entry.opposite_var = string_vars[j][i]
                    entry.bind("<<ComboboxSelected>>",
self.on_combobox_selected)
                    row_entries.append(entry)
                else:
                    var = string_vars[i][j]
                    entry = ttk.Entry(table_frame,
                                      textvariable=var,
                                      width=5,
                                      state="readonly",
                                      style="Custom.TEntry")
                    entry.grid(row=i+1, column=j+1, padx=2, pady=2)
                    row_entries.append(entry)
            matrix_entries.append(row_entries)
        def on_combobox_selected(self, event):
            combobox = event.widget
            value = combobox.get()
            if value.startswith('1/'):
                try:
                    denominator = value[2:]  # Видаляємо "1/"
                    reciprocal = denominator
                except:
                    reciprocal = "1"
            else:
                try:
                    reciprocal = f"1/{value}"
                except:
                    reciprocal = "1"
```

```
    if hasattr(combobox, 'opposite_var'):
        combobox.opposite_var.set(reciprocal)
def save_comparison(self, matrix_type):
    is_criteria = (matrix_type == 'criteria')
    matrix_entries = self.criteria_entries if is_criteria else self.alt_entries
    size = len(matrix_entries)
    matrix = []
    for i in range(size):
        row = []
        for j in range(size):
            cell = matrix_entries[i][j]
            value = 1  # Default value
            if isinstance(cell, ttk.Label):
                value = 1
            elif isinstance(cell, ttk.Entry) and hasattr(cell, 'textvariable'):
                value_str = cell.textvariable.get()
                if value_str.startswith('1/'):
                    try:
                        value = 1 / float(value_str[2:])
                    except:
                        value = 1
                else:
                    try:
                        value = float(value_str)
                    except:
                        value = 1
            elif hasattr(cell, 'get'):
                value_str = cell.get()
                if value_str.startswith('1/'):
                    try:
```

```
                        value = 1 / float(value_str[2:])
                    except:
                        value = 1
                else:
                    try:
                        value = float(value_str)
                    except:
                        value = 1
                row.append(value)
            matrix.append(row)
        weights, cr = self.ahp.calculate_weights(matrix)
        if cr > 0.4:
            messagebox.showwarning("Попередження",
                                   f"Індекс узгодженості CR = {cr:.3f}, що перевищує допустиме значення 0.4. " +
                                   "Рекомендується переглянути оцінки.")
        self.db.save_comparison(self.current_user['username'], matrix_type, matrix, cr)
        messagebox.showinfo("Успіх", "Порівняння збережено")
    def setup_admin_interface(self):
        self.clear_window()
        screen_width = self.root.winfo_screenwidth()
        screen_height = self.root.winfo_screenheight()
        x = (screen_width - 860) // 2
        y = (screen_height - 790) // 2
        self.root.geometry(f"860x790+{x}+{y}")
        self.admin_frame = ttk.Frame(self.root, padding=10)
        self.admin_frame.pack(fill=tk.BOTH, expand=True)
        ttk.Label(self.admin_frame, text="Панель адміністратора", font=('Helvetica', 14)).pack(pady=10)
```

```
    tab_frame = CustomTabFrame(self.admin_frame)
    tab_frame.pack(fill=tk.BOTH, expand=True, pady=10)
    users_tab = ttk.Frame(tab_frame)
    antivirus_tab = ttk.Frame(tab_frame)
    analysis_tab = ttk.Frame(tab_frame)
    tab_frame.add_tab("Користувачі", users_tab)
    tab_frame.add_tab("Антивіруси", antivirus_tab)
    tab_frame.add_tab("Аналіз даних", analysis_tab)
    self.setup_experts_management(users_tab)
    self.setup_antivirus_management(antivirus_tab)
    self.setup_data_analysis(analysis_tab)
    tk.Button(self.admin_frame,
            text="Вийти",
            command=self.setup_login_frame,
            bg='#007bff',
            fg='white',
            activebackground='#0056b3',
            activeforeground='white',
            font=('Segoe UI', 10),
            relief='raised',
            padx=10,
            pady=5).pack(pady=10)
def setup_experts_management(self, parent):
    container = ttk.Frame(parent)
    container.pack(fill=tk.BOTH, expand=True)
    header_frame = ttk.Frame(container)
    header_frame.pack(fill=tk.X, pady=10)
    create_custom_label(header_frame,
            text="Управління користувачами",
            font_size=12,
```

```
                is_bold=True).pack()
        table_frame = ttk.Frame(container)
        table_frame.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)
        self.update_users_list(table_frame)
        button_frame = ttk.Frame(container)
        button_frame.pack(fill=tk.X, pady=10)
        create_custom_button(button_frame,
                             text="Додати користувача",
                             command=self.add_user).pack(side=tk.LEFT, padx=5)
        create_custom_button(button_frame,
                             text="Редагувати користувача",
                             command=self.edit_user).pack(side=tk.LEFT, padx=5)
        create_custom_button(button_frame,
                             text="Видалити користувача",
                             command=self.delete_user,
                             bg_color='#dc3545',
                             hover_color='#c82333').pack(side=tk.LEFT, padx=5)
        create_custom_button(button_frame,
                              text="Оновити список",
                              command=lambda:
self.update_users_list(table_frame)).pack(side=tk.LEFT, padx=5)
    def update_users_list(self, parent):
        for widget in parent.winfo_children():
            widget.destroy()
        users = self.db.get_users()
        if not users:
            if not any(isinstance(w, ttk.Label) for w in parent.winfo_children()):
                ttk.Label(parent, text="Користувачів не знайдено").pack(pady=20)
            return
        table_container = ttk.Frame(parent)
```

```
            table_container.pack(fill=tk.BOTH, expand=True)
            style = ttk.Style()
            style.configure("Export.TButton",
                    background='#007bff',
                    foreground='white',
                    font=('Segoe UI', 8),
                    padding=5)
            self.users_tree = ttk.Treeview(table_container, columns=("username", "role", "status", "actions"), show="headings")
            self.users_tree.heading("username", text="Логін")
            self.users_tree.heading("role", text="Роль")
            self.users_tree.heading("status", text="Статус")
            self.users_tree.heading("actions", text="Дії")
            self.users_tree.column("username", width=150)
            self.users_tree.column("role", width=100)
            self.users_tree.column("status", width=100)
            self.users_tree.column("actions", width=80)
            self.users_tree.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
            scrollbar = ttk.Scrollbar(table_container, orient=tk.VERTICAL, command=self.users_tree.yview)
            scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
            self.users_tree.configure(yscrollcommand=scrollbar.set)
            for user in users:
               if user['role'] == 'expert':
                  criteria_comp = self.db.get_expert_comparison(user['username'], 'criteria')
                  criteria_done = criteria_comp is not None
                  alternatives = self.db.get_antiviruses()
                  alt_comps_done = True
                  for i in range(1, 7):  # K1-K6
```

```
                    alt_comp = self.db.get_expert_comparison(user['username'],
f'alt_by_K{i}')
                    if alt_comp is None:
                        alt_comps_done = False
                        break
                is_completed = criteria_done and alt_comps_done
                status = "Завершено" if is_completed else "Незавершено"
                item = self.users_tree.insert("", tk.END, values=(user['username'],
user['role'], status, "Експорт" if is_completed else "-"))
                if is_completed:
                    self.users_tree.item(item, tags=(user['username'],))
            else:
                self.users_tree.insert("", tk.END, values=(user['username'],
user['role'], "-", "-"))
        self.users_tree.bind('<ButtonRelease-1>', self.handle_export_click)
    def handle_export_click(self, event):
        region = self.users_tree.identify_region(event.x, event.y)
        if region == "cell":
            column = self.users_tree.identify_column(event.x)
            if str(column) == "#4":
                item = self.users_tree.identify_row(event.y)
                if item:
                    values = self.users_tree.item(item)['values']
                    if values[3] == "Експорт":
                        username = self.users_tree.item(item)['tags'][0]
                        self.export_expert_data(username)
    def add_user(self):
        dialog = tk.Toplevel(self.root)
        dialog.title("Додати користувача")
        dialog.geometry("400x350")
```

```
dialog.transient(self.root)
dialog.grab_set()
screen_width = dialog.winfo_screenwidth()
screen_height = dialog.winfo_screenheight()
x = (screen_width - 400) // 2
y = (screen_height - 350) // 2
dialog.geometry(f"400x350+{x}+{y}")
main_frame = ttk.Frame(dialog, padding=20)
main_frame.pack(fill=tk.BOTH, expand=True)
ttk.Label(main_frame,
        text="Додати користувача",
        font=('Segoe UI', 14, 'bold')).pack(pady=(0, 20))
form_frame = ttk.Frame(main_frame)
form_frame.pack(fill=tk.X, pady=10)
ttk.Label(form_frame,
        text="Логін:",
        font=('Segoe UI', 10)).pack(anchor='w', pady=5)
username_var = tk.StringVar()
ttk.Entry(form_frame,
        textvariable=username_var,
        width=30,
        font=('Segoe UI', 10)).pack(fill=tk.X, pady=(0, 10))
ttk.Label(form_frame,
        text="Пароль:",
        font=('Segoe UI', 10)).pack(anchor='w', pady=5)
password_var = tk.StringVar()
ttk.Entry(form_frame,
        textvariable=password_var,
        show="*",
        width=30,
```

```
        font=(‘Segoe UI’, 10)).pack(fill=tk.X, pady=(0, 10))
ttk.Label(form_frame,
        text="Роль:",
        font=(‘Segoe UI’, 10)).pack(anchor=‘w’, pady=5)
role_var = tk.StringVar(value="expert")
role_frame = ttk.Frame(form_frame)
role_frame.pack(fill=tk.X, pady=(0, 10))
style = ttk.Style()
style.configure("Custom.TRadiobutton",
              font=(‘Segoe UI’, 10),
              background=‘#f0f0f0’,
              foreground=‘#333333’)
expert_radio = ttk.Radiobutton(role_frame,
                        text="Експерт",
                        variable=role_var,
                        value="expert",
                        style="Custom.TRadiobutton")
expert_radio.pack(side=tk.LEFT, padx=(0, 20))
admin_radio = ttk.Radiobutton(role_frame,
                       text="Адміністратор",
                       variable=role_var,
                       value="admin",
                       style="Custom.TRadiobutton")
admin_radio.pack(side=tk.LEFT)
button_frame = ttk.Frame(main_frame)
button_frame.pack(fill=tk.X, pady=(20, 0))
def submit():
   username = username_var.get()
   password = password_var.get()
   role = role_var.get()
```

```
            if not username or not password:
                messagebox.showerror("Помилка", "Введіть логін та пароль")
                return
            if self.db.add_user(username, password, role):
                messagebox.showinfo("Успіх", "Користувач доданий")
                dialog.destroy()
                # Оновлюємо тільки таблицю користувачів
                self.update_users_list(self.users_tree.master)
            else:
                messagebox.showerror("Помилка", "Користувач з таким логіном вже існує")
        add_button = tk.Button(button_frame,
                               text="Додати",
                               command=submit,
                               bg='#007bff',
                               fg='white',
                               activebackground='#0056b3',
                               activeforeground='white',
                               font=('Segoe UI', 10),
                               relief='raised',
                               padx=20,
                               pady=5)
        add_button.pack(side=tk.RIGHT)
        cancel_button = tk.Button(button_frame,
                                  text="Скасувати",
                                  command=dialog.destroy,
                                  bg='#e0e0e0',
                                  fg='#333333',
                                  activebackground='#d0d0d0',
                                  activeforeground='#333333',
```

```
                font=('Segoe UI', 10),
                relief='raised',
                padx=20,
                pady=5)
        cancel_button.pack(side=tk.RIGHT, padx=(0, 10))
        main_frame.configure(style='Custom.TFrame')
        form_frame.configure(style='Custom.TFrame')
        role_frame.configure(style='Custom.TFrame')
        button_frame.configure(style='Custom.TFrame')
        dialog.configure(bg='#f0f0f0')
    def delete_user(self):
        selected_item = self.users_tree.selection()
        if not selected_item:
            messagebox.showerror("Помилка", "Виберіть користувача для видалення")
            return
        user_data = self.users_tree.item(selected_item[0])['values']
        username = str(user_data[0])
        role = user_data[1]
        if role == 'admin':
            messagebox.showerror("Помилка", "Неможливо видалити адміністратора")
            return
        if messagebox.askyesno("Підтвердження", f"Ви впевнені, що хочете видалити користувача {username}?"):
            if self.db.delete_user(username):
                messagebox.showinfo("Успіх", "Користувача видалено")
                self.update_users_list(self.users_tree.master)
            else:
```

```
                messagebox.showerror("Помилка", "Не вдалося видалити користувача. Перевірте, чи не є він адміністратором.")
        def edit_user(self):
            selected_item = self.users_tree.selection()
            if not selected_item:
                messagebox.showerror("Помилка", "Виберіть користувача для редагування")
                return
            user_data = self.users_tree.item(selected_item[0])['values']
            username = str(user_data[0])
            role = user_data[1]
            if role == 'admin':
                messagebox.showerror("Помилка", "Неможливо редагувати адміністратора")
                return
            dialog = tk.Toplevel(self.root)
            dialog.title("Редагувати користувача")
            dialog.geometry("400x350")
            dialog.transient(self.root)
            dialog.grab_set()
            screen_width = dialog.winfo_screenwidth()
            screen_height = dialog.winfo_screenheight()
            x = (screen_width - 400) // 2
            y = (screen_height - 350) // 2
            dialog.geometry(f"400x350+{x}+{y}")
            main_frame = ttk.Frame(dialog, padding=20)
            main_frame.pack(fill=tk.BOTH, expand=True)
            ttk.Label(main_frame,
                      text="Редагувати користувача",
                      font=('Segoe UI', 14, 'bold')).pack(pady=(0, 20))
```

```
form_frame = ttk.Frame(main_frame)
form_frame.pack(fill=tk.X, pady=10)
ttk.Label(form_frame,
        text="Логін:",
        font=('Segoe UI', 10)).pack(anchor='w', pady=5)
username_var = tk.StringVar(value=username)
ttk.Entry(form_frame,
        textvariable=username_var,
        width=30,
        font=('Segoe UI', 10)).pack(fill=tk.X, pady=(0, 10))
 ttk.Label(form_frame,
        text="Новий пароль (залиште порожнім, щоб не змінювати):",
        font=('Segoe UI', 10)).pack(anchor='w', pady=5)
password_var = tk.StringVar()
ttk.Entry(form_frame,
        textvariable=password_var,
        show="*",
        width=30,
        font=('Segoe UI', 10)).pack(fill=tk.X, pady=(0, 10))
ttk.Label(form_frame,
        text="Роль:",
        font=('Segoe UI', 10)).pack(anchor='w', pady=5)
role_var = tk.StringVar(value=role)
role_frame = ttk.Frame(form_frame)
role_frame.pack(fill=tk.X, pady=(0, 10))
style = ttk.Style()
style.configure("Custom.TRadiobutton",
            font=('Segoe UI', 10),
            background='#f0f0f0',
            foreground='#333333')
```

```
            expert_radio = ttk.Radiobutton(role_frame,
                                           text="Експерт",
                                           variable=role_var,
                                           value="expert",
                                           style="Custom.TRadiobutton")
            expert_radio.pack(side=tk.LEFT, padx=(0, 20))
            admin_radio = ttk.Radiobutton(role_frame,
                                          text="Адміністратор",
                                          variable=role_var,
                                          value="admin",
                                          style="Custom.TRadiobutton")
            admin_radio.pack(side=tk.LEFT)
            button_frame = ttk.Frame(main_frame)
            button_frame.pack(fill=tk.X, pady=(20, 0))
            def submit():
                new_username = username_var.get()
                new_password = password_var.get()
                new_role = role_var.get()
                if not new_username:
                    messagebox.showerror("Помилка", "Введіть логін")
                    return
                if self.db.update_user(username, new_username, new_password,
new_role):
                    messagebox.showinfo("Успіх", "Користувача оновлено")
                    dialog.destroy()
                    self.update_users_list(self.users_tree.master)
                else:
                    messagebox.showerror("Помилка", "Не вдалося оновити
користувача")
            save_button = tk.Button(button_frame,
```

```
                text="Зберегти",
                command=submit,
                bg='#007bff',
                fg='white',
                activebackground='#0056b3',
                activeforeground='white',
                font=('Segoe UI', 10),
                relief='raised',
                padx=20,
                pady=5)
    save_button.pack(side=tk.RIGHT)
    cancel_button = tk.Button(button_frame,
                  text="Скасувати",
                  command=dialog.destroy,
                  bg='#e0e0e0',
                  fg='#333333',
                  activebackground='#d0d0d0',
                  activeforeground='#333333',
                  font=('Segoe UI', 10),
                  relief='raised',
                  padx=20,
                  pady=5)
    cancel_button.pack(side=tk.RIGHT, padx=(0, 10))
    main_frame.configure(style='Custom.TFrame')
    form_frame.configure(style='Custom.TFrame')
    role_frame.configure(style='Custom.TFrame')
    button_frame.configure(style='Custom.TFrame')
    dialog.configure(bg='#f0f0f0')
def setup_antivirus_management(self, parent):
```

```
        ttk.Label(parent, text="Управління антивірусами", font=('Segoe UI', 12,
'bold')).pack(pady=10)
        antiviruses_frame = ttk.Frame(parent)
        antiviruses_frame.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)
        self.update_antiviruses_list(antiviruses_frame)
        buttons_frame = ttk.Frame(parent)
        buttons_frame.pack(fill=tk.X, pady=10)
        tk.Button(buttons_frame,
            text="Додати антивірус",
            command=self.add_antivirus,
            bg='#007bff',
            fg='white',
            activebackground='#0056b3',
            activeforeground='white',
            font=('Segoe UI', 10),
            relief='raised',
            padx=10,
            pady=5).pack(side=tk.LEFT, padx=5)
        tk.Button(buttons_frame,
            text="Редагувати антивірус",
            command=self.edit_antivirus,
            bg='#007bff',
            fg='white',
            activebackground='#0056b3',
            activeforeground='white',
            font=('Segoe UI', 10),
            relief='raised',
            padx=10,
            pady=5).pack(side=tk.LEFT, padx=5)
        tk.Button(buttons_frame,
```

```
                text="Видалити антивірус",
                command=self.delete_antivirus,
                bg='#dc3545',
                fg='white',
                activebackground='#c82333',
                activeforeground='white',
                font=('Segoe UI', 10),
                relief='raised',
                padx=10,
                pady=5).pack(side=tk.LEFT, padx=5)
        tk.Button(buttons_frame,
                text="Оновити список",
                command=lambda: self.update_antiviruses_list(antiviruses_frame),
                bg='#007bff',
                fg='white',
                activebackground='#0056b3',
                activeforeground='white',
                font=('Segoe UI', 10),
                relief='raised',
                padx=10,
                pady=5).pack(side=tk.LEFT, padx=5)
    def update_antiviruses_list(self, parent):
        for widget in parent.winfo_children():
            widget.destroy()
        antiviruses = self.db.get_antiviruses()
        if not antiviruses:
            ttk.Label(parent, text="Антивірусів не знайдено").pack(pady=20)
            return
        self.antivirus_tree = ttk.Treeview(parent, columns=("name", "cvss"),
show="headings")
```

```
            self.antivirus_tree.heading("name", text="Назва")
            self.antivirus_tree.heading("cvss", text="CVSS")
            self.antivirus_tree.pack(fill=tk.BOTH, expand=True)
            for av in antiviruses:
                self.antivirus_tree.insert("", tk.END, values=(av['name'], av['cvss']),
tags=(str(av['_id']),))
        def add_antivirus(self):
            dialog = tk.Toplevel(self.root)
            dialog.title("Додати антивірус")
            dialog.geometry("400x300")
            dialog.transient(self.root)
            dialog.grab_set()
            screen_width = dialog.winfo_screenwidth()
            screen_height = dialog.winfo_screenheight()
            x = (screen_width - 400) // 2  # Оновлюємо для нового розміру
            y = (screen_height - 300) // 2  # Оновлюємо для нового розміру
            dialog.geometry(f"400x300+{x}+{y}")
            main_frame = ttk.Frame(dialog, padding=20)
            main_frame.pack(fill=tk.BOTH, expand=True)
            ttk.Label(main_frame,
                      text="Додати антивірус",
                      font=('Segoe UI', 14, 'bold')).pack(pady=(0, 20))
            form_frame = ttk.Frame(main_frame)
            form_frame.pack(fill=tk.X, pady=10)
            ttk.Label(form_frame,
                      text="Назва:",
                      font=('Segoe UI', 10)).pack(anchor='w', pady=5)
            name_var = tk.StringVar()
            ttk.Entry(form_frame,
                      textvariable=name_var,
```

```
            width=30,
            font=('Segoe UI', 10)).pack(fill=tk.X, pady=(0, 10))
        ttk.Label(form_frame,
            text="CVSS (0-1):",
            font=('Segoe UI', 10)).pack(anchor='w', pady=5)
        cvss_var = tk.StringVar()
        ttk.Entry(form_frame,
            textvariable=cvss_var,
            width=30,
            font=('Segoe UI', 10)).pack(fill=tk.X, pady=(0, 10))
        button_frame = ttk.Frame(main_frame)
        button_frame.pack(fill=tk.X, pady=(20, 0))
        def submit():
            name = name_var.get()
            cvss_str = cvss_var.get()
            if not name:
                messagebox.showerror("Помилка", "Введіть назву антивірусу")
                return
            try:
                cvss = float(cvss_str)
                if 0 <= cvss <= 1:
                    self.db.db.antivirus.insert_one({'name': name, 'cvss': cvss})
                    messagebox.showinfo("Успіх", "Антивірус доданий")
                    dialog.destroy()
                    self.update_antiviruses_list(self.antivirus_tree.master)
                else:
                    messagebox.showerror("Помилка", "CVSS повинен бути в
діапазоні 0-1")
            except:
```

```
                messagebox.showerror("Помилка", "Неправильний формат
CVSS")
            add_button = tk.Button(button_frame,
                                   text="Додати",
                                   command=submit,
                                   bg='#007bff',
                                   fg='white',
                                   activebackground='#0056b3',
                                   activeforeground='white',
                                   font=('Segoe UI', 10),
                                   relief='raised',
                                   padx=20,
                                   pady=5)
            add_button.pack(side=tk.RIGHT)
            cancel_button = tk.Button(button_frame,
                                      text="Скасувати",
                                      command=dialog.destroy,
                                      bg='#e0e0e0',
                                      fg='#333333',
                                      activebackground='#d0d0d0',
                                      activeforeground='#333333',
                                      font=('Segoe UI', 10),
                                      relief='raised',
                                      padx=20,
                                      pady=5)
            cancel_button.pack(side=tk.RIGHT, padx=(0, 10))
            main_frame.configure(style='Custom.TFrame')
            form_frame.configure(style='Custom.TFrame')
            button_frame.configure(style='Custom.TFrame')
            dialog.configure(bg='#f0f0f0')
```

```
        def edit_antivirus(self):
            selected_item = self.antivirus_tree.selection()
            if not selected_item:
                messagebox.showerror("Помилка", "Виберіть антивірус для редагування")
                return
            av_id = self.antivirus_tree.item(selected_item[0])['tags'][0]
            current_values = self.antivirus_tree.item(selected_item[0])['values']
            current_name = current_values[0]
            current_cvss = current_values[1]
            dialog = tk.Toplevel(self.root)
            dialog.title("Редагувати антивірус")
            dialog.geometry("400x300")
            dialog.transient(self.root)
            dialog.grab_set()
            screen_width = dialog.winfo_screenwidth()
            screen_height = dialog.winfo_screenheight()
            x = (screen_width - 400) // 2  # Оновлюємо для нового розміру
            y = (screen_height - 300) // 2  # Оновлюємо для нового розміру
            dialog.geometry(f"400x300+{x}+{y}")  # Оновлюємо розмір
            main_frame = ttk.Frame(dialog, padding=20)
            main_frame.pack(fill=tk.BOTH, expand=True)
            ttk.Label(main_frame,
                      text="Редагувати антивірус",
                      font=('Segoe UI', 14, 'bold')).pack(pady=(0, 20))
            form_frame = ttk.Frame(main_frame)
            form_frame.pack(fill=tk.X, pady=10)
            ttk.Label(form_frame,
                      text="Назва:",
                      font=('Segoe UI', 10)).pack(anchor='w', pady=5)
```

```
name_var = tk.StringVar(value=current_name)
ttk.Entry(form_frame,
        textvariable=name_var,
        width=30,
        font=('Segoe UI', 10)).pack(fill=tk.X, pady=(0, 10))
ttk.Label(form_frame,
        text="CVSS (0-1):",
        font=('Segoe UI', 10)).pack(anchor='w', pady=5)
cvss_var = tk.StringVar(value=str(current_cvss))
ttk.Entry(form_frame,
        textvariable=cvss_var,
        width=30,
        font=('Segoe UI', 10)).pack(fill=tk.X, pady=(0, 10))
button_frame = ttk.Frame(main_frame)
button_frame.pack(fill=tk.X, pady=(20, 0))
def submit():
    name = name_var.get()
    cvss_str = cvss_var.get()
    if not name:
        messagebox.showerror("Помилка", "Введіть назву антивірусу")
        return
    try:
        cvss = float(cvss_str)
        if 0 <= cvss <= 1:
            self.db.update_antivirus(av_id, name, cvss)
            messagebox.showinfo("Успіх", "Антивірус оновлено")
            dialog.destroy()
            self.update_antiviruses_list(self.antivirus_tree.master)
        else:
```

```
                messagebox.showerror("Помилка", "CVSS повинен бути в діапазоні 0-1")
            except:
                messagebox.showerror("Помилка", "Неправильний формат CVSS")
        save_button = tk.Button(button_frame,
                                text="Зберегти",
                                command=submit,
                                bg='#007bff',
                                fg='white',
                                activebackground='#0056b3',
                                activeforeground='white',
                                font=('Segoe UI', 10),
                                relief='raised',
                                padx=20,
                                pady=5)
        save_button.pack(side=tk.RIGHT)
        cancel_button = tk.Button(button_frame,
                                  text="Скасувати",
                                  command=dialog.destroy,
                                  bg='#e0e0e0',
                                  fg='#333333',
                                  activebackground='#d0d0d0',
                                  activeforeground='#333333',
                                  font=('Segoe UI', 10),
                                  relief='raised',
                                  padx=20,
                                  pady=5)
        cancel_button.pack(side=tk.RIGHT, padx=(0, 10))
        # Встановлюємо фон для всіх фреймів
```

```
            main_frame.configure(style='Custom.TFrame')
            form_frame.configure(style='Custom.TFrame')
            button_frame.configure(style='Custom.TFrame')
            dialog.configure(bg='#f0f0f0')
        def delete_antivirus(self):
            selected_item = self.antivirus_tree.selection()
            if not selected_item:
                messagebox.showerror("Помилка", "Виберіть антивірус для видалення")
                return
            av_id = self.antivirus_tree.item(selected_item[0])['tags'][0]
            av_name = self.antivirus_tree.item(selected_item[0])['values'][0]
            if messagebox.askyesno("Підтвердження", f"Ви впевнені, що хочете видалити антивірус {av_name}?"):
                try:
                    av_id = ObjectId(av_id)
                    if self.db.delete_antivirus(av_id):
                        messagebox.showinfo("Успіх", "Антивірус видалено")
                        self.update_antiviruses_list(self.antivirus_tree.master)
                    else:
                        messagebox.showerror("Помилка", "Не вдалося видалити антивірус")
                except Exception as e:
                    messagebox.showerror("Помилка", f"Помилка при видаленні: {str(e)}")
        def setup_data_analysis(self, parent):
            ttk.Label(parent, text="Аналіз даних та прийняття рішення", font=('Helvetica', 12)).pack(pady=10)
            approach_frame = ttk.Frame(parent)
            approach_frame.pack(fill=tk.X, pady=10)
```

```
        ttk.Label(approach_frame, text="Вибрати підхід:").pack(side=tk.LEFT, padx=5)
        self.selected_approach = tk.StringVar(value="A")
        ttk.Radiobutton(approach_frame, text="Підхід A", variable=self.selected_approach, value="A").pack(side=tk.LEFT, padx=5)
        ttk.Radiobutton(approach_frame, text="Підхід B", variable=self.selected_approach, value="B").pack(side=tk.LEFT, padx=5)
        k_frame = ttk.Frame(approach_frame)
        k_frame.pack(side=tk.LEFT, padx=10)
        ttk.Label(k_frame, text="K для підходу B:").pack(side=tk.LEFT, padx=5)
        self.k_value = tk.StringVar(value="0.5")
        ttk.Entry(k_frame, textvariable=self.k_value, width=5).pack(side=tk.LEFT)
        tk.Button(parent,
            text="Провести аналіз",
            command=self.run_analysis,
            bg='#007bff',
            fg='white',
            activebackground='#0056b3',
            activeforeground='white',
            font=('Segoe UI', 10),
            relief='raised',
            padx=10,
            pady=5).pack(pady=10)
        self.results_frame = ttk.Frame(parent)
        self.results_frame.pack(fill=tk.BOTH, expand=True, pady=10)
    def run_analysis(self):
        for widget in self.results_frame.winfo_children():
            widget.destroy()
        expert_names = self.db.get_expert_names()
```

```
            if not expert_names:
                messagebox.showerror("Помилка", "Немає зареєстрованих експертів")
                return
            incomplete_experts = []
            for expert in expert_names:
                criteria_comp = self.db.get_expert_comparison(expert, 'criteria')
                if not criteria_comp:
                    incomplete_experts.append(f"{expert} (не оцінив критерії)")
                    continue
                for i in range(1, 7):  # К1-К6
                    alt_comp = self.db.get_expert_comparison(expert, f'alt_by_К{i}')
                    if not alt_comp:
                        incomplete_experts.append(f"{expert} (не оцінив альтернативи для критерію К{i})")
                        break
            if incomplete_experts:
                error_message = "Наступні експерти не завершили оцінки:\n\n"
                error_message += "\n".join(incomplete_experts)
                error_message += "\n\nБудь ласка, дочекайтеся завершення всіх оцінок."
                messagebox.showerror("Помилка", error_message)
                return
            antiviruses = self.db.get_antiviruses()
            if not antiviruses:
                messagebox.showerror("Помилка", "Немає антивірусів для аналізу")
                return
            criteria_matrices = []
            for expert in expert_names:
                comparison = self.db.get_expert_comparison(expert, 'criteria')
```

```
            if comparison and ‘matrix’ in comparison:
                criteria_matrices.append(comparison[‘matrix’])
        if not criteria_matrices:
            messagebox.showerror("Помилка", "Немає оцінок критеріїв від експертів")
            return
        aggregated_criteria_matrix = self.ahp.geometric_mean_aggregation(criteria_matrices)
        criteria_weights, criteria_cr = self.ahp.calculate_weights(aggregated_criteria_matrix)
        if criteria_cr > 0.4:
            messagebox.showwarning("Попередження",
                                   f"Загальний індекс узгодженості для критеріїв CR = {criteria_cr:.3f}, " +
                                   "що перевищує допустиме значення 0.4.")
        alt_weights_by_criteria = []
        for i in range(1, 7):  # К1-К6
            alt_matrices = []
            for expert in expert_names:
                comparison = self.db.get_expert_comparison(expert, f’alt_by_К{i}’)
                if comparison and ‘matrix’ in comparison:
                    alt_matrices.append(comparison[‘matrix’])
            if alt_matrices:
                aggregated_alt_matrix = self.ahp.geometric_mean_aggregation(alt_matrices)
                alt_weights, alt_cr = self.ahp.calculate_weights(aggregated_alt_matrix)
                alt_weights_by_criteria.append(alt_weights)
            else:
```

```
                messagebox.showerror("Помилка", f"Немає оцінок альтернатив для критерію K{i}")
                return
            cvss_values = [av['cvss'] for av in antiviruses]
            risk_values = alt_weights_by_criteria[5]
            approach = self.selected_approach.get()
            try:
                k_value = float(self.k_value.get())
                if not (0 <= k_value <= 1):
                    messagebox.showerror("Помилка", "K повинен бути в діапазоні 0-1")
                    return
            except:
                messagebox.showerror("Помилка", "Неправильний формат K")
                return
            if approach == "A":
                results = self.ahp.calculate_approach_a(criteria_weights, alt_weights_by_criteria, cvss_values)
            else:
                results = self.ahp.calculate_approach_b(criteria_weights, alt_weights_by_criteria, risk_values, cvss_values, k_value)
            ttk.Label(self.results_frame, text=f"Результати ({approach})", font=('Helvetica', 12)).pack(pady=10)
            results_table = ttk.Frame(self.results_frame)
            results_table.pack(fill=tk.BOTH, expand=True, padx=10)
            ttk.Label(results_table, text="Антивірус", font=('Helvetica', 10, 'bold')).grid(row=0, column=0, padx=5, pady=5)
            ttk.Label(results_table, text="Оцінка", font=('Helvetica', 10, 'bold')).grid(row=0, column=1, padx=5, pady=5)
```

```
            ttk.Label(results_table, text="Ранг", font=('Helvetica', 10, 'bold')).grid(row=0, column=2, padx=5, pady=5)
            ranked_results = [(antiviruses[i]['name'], results[i], i) for i in range(len(results))]
            ranked_results.sort(key=lambda x: x[1], reverse=True)
            for i, (name, score, idx) in enumerate(ranked_results):
                ttk.Label(results_table, text=name).grid(row=i+1, column=0, padx=5, pady=5)
                ttk.Label(results_table, text=f"{score:.4f}").grid(row=i+1, column=1, padx=5, pady=5)
                ttk.Label(results_table, text=f"{i+1}").grid(row=i+1, column=2, padx=5, pady=5)
            self.visualize_results(ranked_results)
        def visualize_results(self, ranked_results):
            viz_frame = ttk.Frame(self.results_frame)
            viz_frame.pack(fill=tk.BOTH, expand=True, pady=10)
            fig, ax = plt.subplots(figsize=(8, 4))
            names = [result[0] for result in ranked_results]
            scores = [result[1] for result in ranked_results]
            bars = ax.bar(names, scores, color='skyblue')
            ax.set_title('Порівняння антивірусних рішень', fontsize=10)
            ax.set_ylabel('Інтегральна оцінка', fontsize=9)
            ax.set_xlabel('Антивірусне ПЗ', fontsize=9)
            ax.tick_params(axis='both', which='major', labelsize=8)
            for bar, score in zip(bars, scores):
                height = bar.get_height()
                ax.annotate(f'{score:.4f}',
                            xy=(bar.get_x() + bar.get_width() / 2, height),
                            xytext=(0, 3),  # 3 points vertical offset
                            textcoords="offset points",
```

```
                    ha='center', va='bottom', rotation=0,
                    fontsize=8)
            plt.xticks(rotation=45, ha='right')
            plt.tight_layout()
            canvas = FigureCanvasTkAgg(fig, master=viz_frame)
            canvas.draw()
            canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)
        def export_expert_data(self, expert_username):
            try:
                criteria_comp = self.db.get_expert_comparison(expert_username,
'criteria')

                if not criteria_comp or 'matrix' not in criteria_comp:
                    messagebox.showerror("Помилка", f"Немає даних для експорту від
експерта {expert_username}")
                    return
                current_time = datetime.now().strftime("%Y%m%d_%H%M%S")
                filename = f"expert_{expert_username}_{current_time}.xlsx"
                file_path = filedialog.asksaveasfilename(
                    defaultextension=".xlsx",
                    initialfile=filename,
                    title="Зберегти результати експерта",
                    filetypes=[("Excel files", "*.xlsx")]
                )
                if not file_path:
                    return
                wb = openpyxl.Workbook()
                criteria_df = pd.DataFrame(
                    criteria_comp['matrix'],
                    columns=self.criteria_names,
```

```
                index=self.criteria_names
            )
            ws = wb.active
            ws.title = 'Критерії'
            for col_idx, name in enumerate(self.criteria_names, 2):
                ws.cell(row=1, column=col_idx, value=name)
            for row_idx, (name, row) in enumerate(zip(self.criteria_names, criteria_df.values), 2):
                ws.cell(row=row_idx, column=1, value=name)  # Заголовок рядка
                for col_idx, value in enumerate(row, 2):
                    ws.cell(row=row_idx, column=col_idx, value=value)
            antiviruses = self.db.get_antiviruses()
            av_names = [av['name'] for av in antiviruses]
            for i in range(1, 7):  # K1-K6
                alt_comp = self.db.get_expert_comparison(expert_username, f'alt_by_K{i}')
                if alt_comp and 'matrix' in alt_comp:
                    ws = wb.create_sheet(title=f'Критерій K{i}')
                    alt_df = pd.DataFrame(
                        alt_comp['matrix'],
                        columns=av_names,
                        index=av_names
                    )
                    for col_idx, name in enumerate(av_names, 2):
                        ws.cell(row=1, column=col_idx, value=name)
                    for row_idx, (name, row) in enumerate(zip(av_names, alt_df.values), 2):
                        ws.cell(row=row_idx, column=1, value=name)  # Заголовок рядка
                        for col_idx, value in enumerate(row, 2):
```

```
                    ws.cell(row=row_idx, column=col_idx, value=value)
            wb.save(file_path)
            messagebox.showinfo("Успіх", f"Дані експерта {expert_username} успішно експортовано в Excel")
        except Exception as e:
            messagebox.showerror("Помилка", f"Помилка при експорті даних: {str(e)}")
if __name__ == "__main__":
    root = tk.Tk()
    app = AntivirusAnalysisApp(root)
    root.mainloop()
```

Код ahp_calculator.py

```
import numpy as np
from scipy.linalg import eig
import math
class AHPCalculator:
    def __init__(self):
        self.RI = {1: 0, 2: 0, 3: 0.58, 4: 0.9, 5: 1.12, 6: 1.24, 7: 1.32, 8: 1.41, 9: 1.45, 10: 1.49}
    def calculate_weights(self, matrix):
        n = len(matrix)
        A = np.array(matrix, dtype=float)
        eigenvalues, eigenvectors = eig(A)
        max_idx = np.argmax(eigenvalues.real)
        max_eigval = eigenvalues[max_idx].real
        max_eigvec = eigenvectors[:, max_idx].real
        weights = max_eigvec / np.sum(max_eigvec)
        ci = (max_eigval - n) / (n - 1) if n > 1 else 0
        cr = ci / self.RI[n] if n > 1 else 0
        return weights, cr
```

```
        def geometric_mean_aggregation(self, matrices):
            if not matrices:
                return None
            n = len(matrices[0])
            result = np.ones((n, n))
            for i in range(n):
                for j in range(n):
                    product = 1.0
                    for matrix in matrices:
                        product *= matrix[i][j]
                    result[i][j] = product ** (1.0 / len(matrices))
            return result.tolist()
        def calculate_approach_a(self, criteria_weights, alt_weights_by_criteria, cvss_values):
            n_alternatives = len(alt_weights_by_criteria[0])
            results = []
            for i in range(n_alternatives):
                weighted_sum = 0
                for j in range(len(criteria_weights)):
                    weighted_sum += criteria_weights[j] * alt_weights_by_criteria[j][i]
                result = weighted_sum * cvss_values[i]
                results.append(result)
            return results
        def calculate_approach_b(self, criteria_weights, alt_weights_by_criteria, risk_values, cvss_values, k=0.5):
            n_alternatives = len(alt_weights_by_criteria[0])
            results = []
            for i in range(n_alternatives):
                base_sum = 0
                for j in range(5):
```

```
            base_sum += criteria_weights[j] * alt_weights_by_criteria[j][i]
        adjusted = base_sum * (1 - k * risk_values[i])
        result = adjusted * cvss_values[i]
        results.append(result)
    return results
```

Код database_connection.py

```
import pymongo
from pymongo import MongoClient
from password_utils import hash_password, verify_password
from datetime import datetime
class Database:
  def __init__(self):
    try:
      self.client = MongoClient()
      self.db = self.client['antivirus_analysis']
      self.client.admin.command('ping')
      if 'users' not in self.db.list_collection_names():
        print("Створення колекції users")
        self.db.create_collection('users')
        self.db.users.insert_one({
          'username': 'admin',
          'password': hash_password('admin'),
          'role': 'admin'
        })
      if 'antivirus' not in self.db.list_collection_names():
        print("Створення колекції antivirus")
        self.db.create_collection('antivirus')
        self.db.antivirus.insert_many([
          {'name': 'ESET NOD32 Antivirus', 'cvss': 0.84},
          {'name': 'Bitdefender Antivirus', 'cvss': 0.8},
```

```
                {‘name’: ‘Avast Business Antivirus’, ‘cvss’: 0.65},
                {‘name’: ‘Check Point Endpoint Security’, ‘cvss’: 0.68},
                {‘name’: ‘McAfee Total Protection’, ‘cvss’: 0.72},
                {‘name’: ‘Zillya! Антивірус’, ‘cvss’: 0.78}
            ])
        if ‘comparisons’ not in self.db.list_collection_names():
            print("Створення колекції comparisons")
            self.db.create_collection(‘comparisons’)
        try:
            self.db.comparisons.drop_index(‘expert_1_type_1’)
        except:
            pass
        # Створюємо індекси
        self.db.users.create_index([(‘username’, 1)], unique=True)
        self.db.antivirus.create_index([(‘name’, 1)], unique=True)
        self.db.comparisons.create_index([(‘expert’, 1), (‘matrix_type’, 1)],
unique=True, sparse=True, name=‘expert_1_matrix_type_1’)
    except Exception as e:
        print(f"Помилка при ініціалізації бази даних: {str(e)}")
        raise
def get_users(self):
    return list(self.db.users.find({}))
def get_user(self, username, password):
    user = self.db.users.find_one({‘username’: username})
    if user and verify_password(password, user[‘password’]):
        return user
    return None
def add_user(self, username, password, role=‘expert’):
    try:
        self.db.users.insert_one({
```

```
                'username': username,
                'password': hash_password(password),
                'role': role
            })
            return True
        except:
            return False
    def update_user(self, old_username, new_username, new_password=None, new_role=None):
        try:
            self.client.admin.command('ping')
            with self.client.start_session() as session:
                with session.start_transaction():
                    user = self.db.users.find_one({'username': old_username})
                    if not user:
                        print(f"Користувач {old_username} не знайдений")
                        return False
                    if old_username != new_username:
                        existing_user = self.db.users.find_one({'username': new_username})
                        if existing_user:
                            print(f"Користувач з ім'ям {new_username} вже існує")
                            return False
                    if user['role'] == 'admin' and new_role and new_role != 'admin':
                        print("Не можна змінити роль адміністратора")
                        return False
                    update_data = {'username': new_username}
                    if new_password:
                        update_data['password'] = hash_password(new_password)
                    if new_role:
```

```
                    update_data['role'] = new_role
                result = self.db.users.update_one(
                    {'username': old_username},
                    {'$set': update_data}
                )
                if result.modified_count == 0:
                    print("Не вдалося оновити користувача")
                    return False
                if old_username != new_username:
                    if user['role'] == 'expert':
                        update_result = self.db.comparisons.update_many(
                            {'expert': old_username},
                            {'$set': {'expert': new_username}}
                        )
                        print(f"Оновлено {update_result.modified_count} порівнянь
для експерта {old_username}")

                    return True
            except Exception as e:
                print(f"Помилка при оновленні користувача: {str(e)}")
                return False
        def get_antiviruses(self):
            return list(self.db.antivirus.find({}))
        def save_comparison(self, expert, matrix_type, matrix, cr):
            try:
                self.client.admin.command('ping')
                with self.client.start_session() as session:
                    try:
                        with session.start_transaction():
                            existing = self.db.comparisons.find_one({
```

```
                'expert': expert,
                'matrix_type': matrix_type
            }, session=session)
            if existing:
                result = self.db.comparisons.update_one(
                    {
                        'expert': expert,
                        'matrix_type': matrix_type
                    },
                    {
                        '$set': {
                            'matrix': matrix,
                            'cr': cr,
                            'last_updated': datetime.now()
                        }
                    },
                    session=session
                )
                if result.modified_count == 0:
                    print(f"Попередження: Не вдалося оновити порівняння для {expert}, тип {matrix_type}")
                    session.abort_transaction()
                    return False
            else:
                result = self.db.comparisons.insert_one({
                    'expert': expert,
                    'matrix_type': matrix_type,
                    'matrix': matrix,
                    'cr': cr,
                    'created_at': datetime.now(),
```

```
                    'last_updated': datetime.now()
                }, session=session)
                if not result.inserted_id:
                    print(f"Попередження: Не вдалося вставити порівняння для {expert}, тип {matrix_type}")
                    session.abort_transaction()
                    return False
                saved = self.db.comparisons.find_one({
                    'expert': expert,
                    'matrix_type': matrix_type
                }, session=session)
                if saved:
                    session.commit_transaction()
                    return True
                else:
                    print(f"Помилка: Порівняння не збережено для {expert}, тип {matrix_type}")
                    session.abort_transaction()
                    return False
            except Exception as e:
                print(f"Помилка в транзакції: {str(e)}")
                session.abort_transaction()
                return False
        except pymongo.errors.ConnectionFailure:
            print("Помилка: Не вдалося підключитися до бази даних")
            return False
        except pymongo.errors.OperationFailure as e:
            print(f"Помилка операції з базою даних: {e}")
            return False
        except Exception as e:
```

```
            print(f"Невідома помилка при збереженні порівняння: {e}")
            return False
    def get_expert_comparison(self, expert, matrix_type):
        try:
            self.client.admin.command('ping')
            with self.client.start_session() as session:
                try:
                    with session.start_transaction():
                        comparison = self.db.comparisons.find_one({
                            'expert': expert,
                            'matrix_type': matrix_type
                        }, session=session)
                        if comparison:
                            session.commit_transaction()
                            return comparison
                        else:
                            session.commit_transaction()
                            return None
                except Exception as e:
                    print(f"Помилка в транзакції: {str(e)}")
                    session.abort_transaction()
                    return None
        except pymongo.errors.ConnectionFailure:
            print("Помилка: Не вдалося підключитися до бази даних")
            return None
        except pymongo.errors.OperationFailure as e:
            print(f"Помилка операції з базою даних: {e}")
            return None
        except Exception as e:
            print(f"Невідома помилка при отриманні порівняння: {e}")
```

```
        return None
    def get_all_comparisons(self):
        return list(self.db.comparisons.find({}))
    def get_expert_names(self):
        experts = self.db.users.find({‘role’: ‘expert’})
        return [expert[‘username’] for expert in experts]
    def update_antivirus(self, av_id, new_name, new_cvss):
        self.db.antivirus.update_one(
            {‘_id’: av_id},
            {‘$set’: {‘name’: new_name, ‘cvss’: new_cvss}}
        )
    def delete_user(self, username):
        try:
            user = self.db.users.find_one({‘username’: username})
            if not user:
                return False
            if user.get(‘role’) == ‘admin’:
                return False
            result = self.db.users.delete_one({‘username’: username})
            self.db.comparisons.delete_many({‘expert’: username})
            return result.deleted_count > 0
        except Exception as e:
            return False
    def delete_antivirus(self, antivirus_id):
        try:
            result = self.db.antivirus.delete_one({‘_id’: antivirus_id})
            return result.deleted_count > 0
        except Exception as e:
            print(f"Помилка при видаленні антивірусу: {e}")
            return False
```

Код password_utils.py

```
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
import base64
import os
def hash_password(password: str) -> str:
    salt = os.urandom(16)
    kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        length=32,
        salt=salt,
        iterations=100000,
    )
    key = kdf.derive(password.encode())
    final_hash = salt + key
    return base64.b64encode(final_hash).decode('utf-8')
def verify_password(password: str, hashed_password: str) -> bool:
    try:
        decoded = base64.b64decode(hashed_password.encode('utf-8'))
        salt = decoded[:16]
        stored_key = decoded[16:]
        kdf = PBKDF2HMAC(
            algorithm=hashes.SHA256(),
            length=32,
            salt=salt,
            iterations=100000,
        )
        key = kdf.derive(password.encode())
        return key == stored_key
    except:
```

```
        return False
```

Код styles.py

```
import tkinter as tk
from tkinter import ttk
def setup_custom_styles():
    style = ttk.Style()
    style.configure(".",
                    font=('Segoe UI', 10),
                    background='#f0f0f0',
                    foreground='#333333')
    style.configure("Title.TLabel",
                    font=('Segoe UI', 16, 'bold'),
                    padding=10,
                    background='#f0f0f0',
                    foreground='#333333')
    style.configure("Custom.TEntry",
                    padding=5,
                    fieldbackground='white',
                    foreground='#333333')
    style.configure("Custom.TCombobox",
                    padding=5,
                    fieldbackground='white',
                    foreground='#333333',
                    selectbackground='#007bff',
                    selectforeground='white')
    style.configure("Treeview",
                    font=('Segoe UI', 10),
                    rowheight=25,
                    background='white',
                    fieldbackground='white',
```

```
                foreground='#333333')
        style.configure("Treeview.Heading",
                font=('Segoe UI', 10, 'bold'),
                padding=5,
                background='#f0f0f0',
                foreground='#333333')
        style.configure("Custom.TRadiobutton",
                font=('Segoe UI', 10),
                background='#f0f0f0',
                foreground='#333333')
        style.configure("Custom.TFrame",
                background='#f0f0f0')
        style.configure("Custom.TNotebook",
                background='#f0f0f0')
        style.configure("Custom.TNotebook.Tab",
                background='#e0e0e0',
                foreground='#333333',
                padding=[10, 5],
                font=('Segoe UI', 10))
        style.map("Custom.TNotebook.Tab",
              background=[('selected', '#007bff')],
              foreground=[('selected', 'white')])
    def create_custom_button(master, text, command, bg_color='#007bff',
hover_color='#0056b3'):
        return tk.Button(master,
                 text=text,
                 command=command,
                 bg=bg_color,
                 fg='white',
                 activebackground=hover_color,
```

```
                activeforeground='white',
                font=('Segoe UI', 10),
                relief='raised',
                padx=10,
                pady=5)
def create_custom_label(master, text, font_size=10, is_bold=False):
    font = ('Segoe UI', font_size)
    if is_bold:
        font = ('Segoe UI', font_size, 'bold')
    return ttk.Label(master,
                text=text,
                font=font,
                style="Custom.TLabel")
```

ДОДАТОК В
СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Копія наукової публікації

Бучик С.С., Маєвський М.О. Система вибору антивірусного програмного забезпечення для об’єктів критичної інфраструктури. VIII Міжнародна науково-практична конференція «Проблеми кібербезпеки інформаційно-комунікаційних систем (PCSICS)». – Київ 2025. – С. 91-92

Система вибору антивірусного програмного забезпечення для об’єктів критичної інфраструктури

Бучик Сергій [1], Маєвський Максим [2]

[1] Кафедра кібербезпеки, Київський Національний університет імені Тараса Шевченка, Україна,
м.Київ, вул.Б.Гаврилишина, 24,
E-mail: buchyk@knu.ua

[2] Кафедра кібербезпеки, Київський Національний університет імені Тараса Шевченка, Україна,
м.Київ, вул.Б.Гаврилишина, 24,
E-mail: maievskyim@knu.ua

Abstract – Strong defense systems for critical information systems become essential because information technology evolves quickly and cyber-attacks grow more complex. The study introduces a method to choose antivirus software designed specifically for critical infrastructure. The research methodology combines existing approach analysis with the creation of evaluation criteria that cover detection

probability and false positive rate and system load and response time and update frequency and operational risk followed by multi-criteria analysis framework design. The expert system decision-making process receives support from an integrated architectural model which guarantees objective and adaptable decision automation.

Ключові слова – антивірусне програмне забезпечення, критична інфраструктура, багатокритеріальний аналіз, кібербезпека, експертна система, операційний ризик.

Вступ

Сучасний розвиток інформаційних технологій супроводжується збільшенням кількості та складності кібератак, що висуває високі вимоги до захисту критично важливих інформаційних систем [1].

Антивірусні рішення повинні демонструвати свою здатність швидко виявляти нові загрози, відповідати нормативним вимогам кібербезпеки та забезпечувати стабільну роботу критичних інфраструктур [2].

Аналіз існуючих підходів та проблем вибору антивірусного програмного забезпечення

Ринок антивірусного програмного забезпечення характеризується сильним динамізмом. Однак традиційні методи, засновані виключно на аналізі сигнатур, часто не можуть виявити поліморфні віруси та атаки нульового дня [1]. Сучасні технології вимагають інтеграції класичних підходів з евристичним аналізом і машинним навчанням, що дозволяє врахувати особливості функціонування критичної інфраструктури інформаційних систем. Недостатня адаптивність існуючих рішень і відсутність уніфікованої методології оцінки породжують необхідність розробки нових підходів, здатних забезпечити високий рівень захисту.

Аналіз поточних підходів показує, що проблема не обмежується лише технологічними аспектами, але й включає організаційні та нормативні виклики. Успішна інтеграція нових технологій вимагає комплексного підходу, що поєднує технічні інновації з впровадженням ефективних управлінських стратегій для забезпечення стабільності та безперервності роботи критичних систем.

Динаміка змін у кіберсередовищі, зумовлена глобалізацією та появою нових методів атаки, вимагає використання сучасних алгоритмів аналізу даних та інтеграції систем штучного інтелекту. Це дозволяє не лише покращити швидкість і точність виявлення загроз, але й зменшити потенційні фінансові та операційні втрати при виникненні інцидентів [1].

Формування критеріїв оцінки антивірусного програмного забезпечення

Об’єктивний вибір засобу захисту передбачає розробку системи критеріїв, що охоплюють як технічні, так і експлуатаційні аспекти.

Ключові параметри включають:

- імовірність виявлення шкідливого коду: визначає здатність системи розпізнавати відомі та нові загрози;

- рівень помилкових позитивних результатів: характеризує точність алгоритмів, мінімізуючи зайві сповіщення;

- завантаження системи та середній час відповіді: вони дозволяють оцінити вплив антивірусного програмного забезпечення на загальну продуктивність інформаційної системи;

- частота оновлення бази даних: забезпечує постійну актуальність інформації про останні загрози;

- інтеграція операційного ризику (фактор ризику): враховує потенційні загрози, пов’язані з роботою засобів захисту в умовах високої важливості для системи;

- відповідність державним стандартам і нормативним вимогам: забезпечує відповідність обраного рішення законодавчій базі [2].

Критерії базуються на попарному порівнянні експертних оцінок із застосуванням середнього геометричного, що дозволяє зменшити вплив суб’єктивних факторів.

Розробка методики порівняльного аналізу антивірусних рішень

Метод багатокритеріального аналізу дозволяє об’єктивно порівнювати антивірусні рішення за такими етапами:

1. Побудова попарних матриць: експерти порівнюють продукти за кожним із заданих критеріїв для отримання вагових коефіцієнтів.

2. Агрегування експертних оцінок: результати індивідуальних порівнянь об’єднуються з використанням середнього геометричного, що мінімізує суб’єктивне упередження.

3. Інтеграція коефіцієнтів ризику: використовуються два підходи: множення суми критеріальних оцінок на фіксований коефіцієнт безпеки або застосування поправочного коефіцієнта після розрахунку базової оцінки, що дає змогу отримати загальний показник ефективності.

4. Перевірка узгодженості оцінок: розрахунок індексу узгодженості (ІУ) гарантує достовірність отриманих даних; при перевищенні встановленого порогу проводиться додаткове обстеження.

1. Особлива увага приділяється підбору засобів, включених до переліку ДССЗЗІ, що забезпечує відповідність нормативним вимогам і високий рівень захисту [3].

Аналіз методики демонструє, що інтеграція багатокритеріального підходу дозволяє не лише об’єктивно порівнювати різні засоби захисту, але й адаптувати систему оцінки до змін у кіберсередовищі. Це забезпечує гнучкість моделі та її здатність реагувати на нові виклики в галузі кібербезпеки.

Розробка архітектури експертної системи для автоматизації вибору антивірусного програмного забезпечення

Архітектура експертної системи включає ключові компоненти, що забезпечують автоматизацію процесу прийняття рішень:

1. Модуль збору даних: систематично збирає інформацію про технічні характеристики та робочі показники засобів захисту, створюючи унікальну базу даних для аналізу.

2. Модуль алгоритму побудови парної матриці: автоматизовано створення порівняльних таблиць за всіма заданими критеріями з використанням середнього геометричного для розрахунку вагових коефіцієнтів.

3. Модуль інтеграції індикатора ризику: реалізує альтернативні підходи до інтеграції факторів ризику, забезпечуючи загальний індикатор ефективності кожного антивірусного рішення та забезпечуючи гнучкість, що дозволяє моделі адаптуватися до різних умов роботи.

4. Система контролю узгодженості: контролює достовірність експертних оцінок шляхом розрахунку індексу узгодженості (ІУ), що дозволяє виявити невідповідності та розпочати більш глибокий аналіз.

5. Інтерфейс користувача: забезпечує зручний доступ до результатів сканування, відображає оцінку засобів захисту та пропонує рекомендації для прийняття зваженого рішення.

Архітектура, розроблена на основі сучасних алгоритмічних підходів, може бути інтегрована в існуючі системи управління інформаційною безпекою для підвищення об’єктивності у виборі оптимальних засобів захисту [3, 4, 5].

Дана структура дозволяє розширювати функціональність експертної системи шляхом інтеграції нових модулів аналізу даних, що забезпечує її адаптивність до змін у кібербезпеці та постійне вдосконалення процесу прийняття рішень.

Висновки

Методологія вибору антивірусного програмного забезпечення, яка поєднує сучасний аналіз кіберзагроз, розробку єдиної системи критеріїв оцінки, використання багатокритеріального аналізу та автоматизацію процесів за допомогою експертної системи, дозволяє знизити операційні ризики та оптимізувати витрати на кібербезпеку. Система підтримки прийняття рішень сприяє впровадженню інноваційних технологій кібербезпеки, забезпечуючи обґрунтований та ефективний вибір інструментів захисту в умовах цифрового середовища, що швидко розвивається.

Інтегроване рішення відкриває нові перспективи для подальшого розвитку систем захисту, що відповідають сучасним вимогам та дозволяють адаптуватися до постійно зростаючих викликів кібербезпеки.

Література

[1] Giannopoulos G, Filippini R, Schimmer M. Risk assessment methodologies for Critical Infrastructure Protection. Part I: A state of the art. EUR 25286 EN. Luxembourg (Luxembourg): Publications Office of the European Union; 2012. JRC70046. DOI: https://dx.doi.org/10.2788/22260

[2] Про Стратегію кібербезпеки України [Електронний ресурс]: Указ Президента України від 26.08.2021 №447/2021 – Режим доступу: https://zakon.rada.gov.ua/laws/show/447/2021#Text

[3] Засоби ТЗІ, які мають експертний висновок про відповідність вимогам технічного захисту інформації [Електронний ресурс] – Режим доступу: https://cip.gov.ua/ua/news/zasobi-tzi-yaki-mayut-ekspertnii-visnovok-pro-vidpovidnist-do-vimog-tekhnichnogo-zakhistu-informaciyi

[4] Herzog, C., Tong, V. V. T., Wilke, P., Van Straaten, A. and Lanet, J.-L. (2020). Evasive Windows Malware: Impact on Antiviruses and Possible Countermeasures. In Proceedings of the 17th International Joint Conference on e-

Business and Telecommunications - SECRYPT; ISBN 978-989-758-446-6; ISSN 2184-7711, SciTePress, pages 302-309. DOI: 10.5220/0009816703020309

[5] Про основні засади забезпечення кібербезпеки України [Електронний ресурс]: Закон України від 05.10.2017 № 2163-VIII – Режим доступу: https://zakon.rada.gov.ua/laws/show/2163-19#Text