МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ
ЛЬВІВСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ВНУТРІШНІХ СПРАВ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ УПРАВЛІННЯ, ПСИХОЛОГІЇ ТА БЕЗПЕКИ

Кафедра інформаційних технологій

РОЗРОБЛЕННЯ ПРОГРАМИ-СКАНЕРА БЕЗДРОТОВИХ КОМП'ЮТЕРНИХ МЕРЕЖ ДЛЯ ПЕРЕВІРКИ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ ДЕРЖАВНИХ ОБ'ЄКТІВ

кваліфікаційна робота
здобувача вищої освіти
4 курсу заочної форми навчання
Назара ФІЯЛКИ

Науковий керівник:
Михайло РИЛЬНІКОВ

Рецензент:

_____
вчене звання, науковий ступінь

_____
(Ім'я ПРІЗВИЩЕ рецензента)

Кваліфікаційна робота допущена до захисту
«___» _____ 2026 р., протокол № _

Завідувач кафедри інформаційних технологій
_____ **Олег ЗАЧЕК**
(підпис)

Львів
2026

СПИСОК УМОВНИХ СКОРОЧЕНЬ

CPU	Central Processing Unit (центральний процесор ЕОМ)
SQL	Structured query language (мова структурованих запитів)
OS (ОС)	Operating System (операційна система)
RAM	Random Access Memory (оперативна пам'ять ЕОМ)
UI/UX	User Interface / User Experience (інтерфейс користувача та досвід взаємодії)
REST	Representational State Transfer (архітектурний стиль взаємодії компонентів розподіленого застосунку)
SPA	Single Page Application (односторінковий веб-застосунок)
API	Application Programming Interface (інтерфейс програмування застосунків)
PDF	Portable Document Format (кросплатформний формат електронних документів для звітів)
АРМ	Автоматизоване робоче місце
БД	База даних
КЗІ	Комплексний захист інформації
ІС	Інформаційна система
ЕОМ	Електронна обчислювальна машина
ТЗІ	Технічний захист інформації
ПЗ	Програмне забезпечення
САС	Системи аналітичного супроводження
АС	Автоматизована система
СУБД	Система управління базами даних

АНОТАЦІЯ

ФІЯЛКА Н. Розроблення програми-сканера бездротових комп'ютерних мереж для перевірки інформаційної безпеки державних об'єктів. – Рукопис.

Дослідження на здобуття освітнього ступеня «бакалавр» за спеціальністю 126 «Інформаційні системи та технології». – Львівський державний університет внутрішніх справ, МВС України, Львів, 2026.

У роботі розроблено програму-сканер, призначену для аналізу стану інформаційної безпеки бездротових комп'ютерних мереж на об'єктах державної власності. Проведено аналіз наявних інструментів та обґрунтовано вибір технологічного стеку. Реалізовано модульну архітектуру програмного комплексу та підсистему генерації технічних PDF-звітів. Результатом дослідження є функціональний прототип системи.

Ключові слова: інформаційна безпека, технічний захист інформації, програма-сканер, бездротові мережі, державні об'єкти, оцінка ризиків.

ABSTRACT

FIIALKA N. Development of a wireless computer network scanner program for verifying information security of state objects. – Manuscript.

Research for obtaining a bachelor’s degree in specialty 126 «Information systems and technologies». – Lviv State University of Internal Affairs, MIA of Ukraine, Lviv, 2026.

The work presents the development of a scanner program designed for analysis of the information security status of wireless computer networks at state-owned objects. Existing tools were analyzed, and the choice of technology stack. A modular architecture of the software package and a subsystem for generation of technical PDF reports were implemented. The result of the study is a functional system prototype.

Keywords: information security, technical protection of information, scanner program, wireless networks, state objects, risk assessment.

ЗМІСТ

ВСТУП.....6

РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМИ БЕЗПЕКИ БЕЗДРОТОВИХ МЕРЕЖ НА ДЕРЖАВНИХ ОБ'ЄКТАХ.....9

1.1. Особливості нормативно-правового регулювання захисту інформації в держустановах.....9

1.2. Огляд існуючих програмних засобів моніторингу.....11

1.3. Порівняльний аналіз аналогів та обґрунтування розробки власного рішення.....14

РОЗДІЛ 2 ПРОЄКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ МОНІТОРИНГУ.....17

2.1. Визначення функціональних та технічних вимог до системи.....17

2.2. Обґрунтування вибору архітектури та технологічного стеку.....20

2.3. Розробка алгоритму пасивного сканування та кросплатформної взаємодії з ОС.....23

2.4. Методика оцінки рівня безпеки мережі (визначення критеріїв ризику).....25

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ СКАНЕРА БЕЗДРОТОВИХ МЕРЕЖ.....29

3.1. Структура програмного комплексу та взаємодія модулів.....29

3.2. Розробка графічного інтерфейсу користувача (GUI)......32

3.3. Реалізація підсистеми зберігання історії та генерації PDF-звітів....37

РОЗДІЛ 4 ТЕСТУВАННЯ ТА АНАЛІЗ ЕФЕКТИВНОСТІ РОБОТИ ПРОГРАМИ.....42

4.1. Сценарії тестування в різних операційних середовищах.....42

4.2. Аналіз результатів виявлення вразливостей (відкриті мережі, застарілі протоколи).....44

4.3. Оцінка швидкодії та стабільності роботи під навантаженням.....47

ВИСНОВКИ.....50

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....53

ДОДАТКИ.....55

ВСТУП

Актуальність теми. Процеси цифровізації та впровадження новітніх інформаційних технологій у діяльність державних органів влади та об’єктів критичної інфраструктури України супроводжуються зростанням ризиків для інформаційної безпеки. В умовах постійних кіберзагроз та гібридної агресії, захист периметра державних установ стає одним із пріоритетних завдань національної безпеки. Одним із найбільш вразливих сегментів інформаційно-комунікаційних систем є бездротові мережі стандарту IEEE 802.11 (Wi-Fi).

Бездротові технології, попри свою зручність та мобільність, мають специфічні архітектурні вразливості, зумовлені природою передачі сигналу через відкрите середовище. Для державних об’єктів, де циркулює інформація з обмеженим доступом, загрози перехоплення трафіку, розгортання несанкціонованих точок доступу (Rogue AP) або використання застарілих протоколів шифрування (WEP, WPA) можуть призвести до критичних витоків даних.

Існуючі комерційні та відкриті програмні засоби моніторингу часто мають або надто вузьку спеціалізацію, або використовують активні методи сканування, які можуть демаскувати процес перевірки або створювати завади для роботи спеціалізованого обладнання. Таким чином, існує об’єктивна потреба у розробленні вітчизняного програмного інструментарію для пасивного сканування, який дозволяв би проводити комплексний аудит безпеки ефіру без активного втручання в роботу мереж, забезпечуючи при цьому високу мобільність та кросплатформність.

Розроблення програми-сканера дозволить автоматизувати процес виявлення небезпечних налаштувань та несанкціонованих пристроїв, що є важливим кроком у побудові комплексної системи захисту інформації (КСЗІ) на державних об’єктах.

Мета роботи - розроблення кросплатформного програмного комплексу для пасивного виявлення, аналізу та оцінки стану безпеки бездротових комп'ютерних мереж на об'єктах державної власності з подальшою автоматизацією формування звітної документації.

Для досягнення поставленої мети необхідно вирішити такі **завдання**:

1. Проаналізувати сучасний стан безпеки бездротових технологій та нормативно-правові вимоги щодо захисту інформації в державних установах України.
2. Провести порівняльний аналіз існуючих засобів моніторингу Wi-Fi мереж та визначити критерії для створення власного рішення.
3. Розробити алгоритм пасивного збору технічних параметрів бездротових мереж у кросплатформному середовищі.
4. Спроектувати та реалізувати методику автоматизованої оцінки рівнів безпеки точок доступу на основі сукупності виявлених ознак (шифрування, потужність сигналу, SSID).
5. Створити програмний інтерфейс користувача для візуалізації результатів сканування та підсистему зберігання історії в локальній базі даних.
6. Реалізувати модуль генерації аналітичних PDF-звітів, що відповідають вимогам діловодства в державних структурах.

Об'єкт дослідження - процеси моніторингу та оцінювання стану інформаційної безпеки бездротових комп'ютерних мереж.

Предмет дослідження - методи, алгоритми та програмні засоби пасивного виявлення вразливостей Wi-Fi мереж на державних об'єктах.

Методи дослідження. У роботі використано методи системного аналізу для вивчення архітектури мереж та загроз, принципи об'єктно - орієнтованого програмування для побудови структури програми, а також методи математичного моделювання для розробки системи оцінки ризиків.

Наукова новизна одержаних результатів полягає у вдосконаленні алгоритму пасивної ідентифікації прихованих загроз бездротового ефіру

шляхом комбінування аналізу технічних параметрів сигналу та кросплатформної взаємодії з системними утилітами ОС без необхідності використання спеціалізованих драйверів у режимі моніторингу.

Практичне значення роботи полягає у створенні автономного програмного інструменту, який може бути використаний офіцерами безпеки та системними адміністраторами державних установ для оперативного аудиту бездротового простору, виявлення «тіньових» ІТ-ресурсів та підготовки документації щодо стану захищеності об’єкта.

РОЗДІЛ 1

АНАЛІЗ ПРОБЛЕМИ БЕЗПЕКИ БЕЗДРОТОВИХ МЕРЕЖ НА ДЕРЖАВНИХ ОБ'ЄКТАХ

1.1. Особливості нормативно-правового регулювання захисту інформації в держустановах

Забезпечення інформаційної безпеки в державних установах України є стратегічним завданням, що регулюється низкою законодавчих та нормативно-правових актів. Оскільки державні об’єкти оперують інформацією, що є власністю держави, або інформацією з обмеженим доступом, будь-які технічні рішення, що впроваджуються в таких установах, повинні відповідати вимогам законодавства у сфері технічного захисту інформації (ТЗІ).

Основоположним актом у цій сфері є Закон України «Про захист інформації в інформаційно-комунікаційних системах». Згідно зі статтею 8 цього Закону, державні інформаційні ресурси або інформація з обмеженим доступом повинні оброблятися в системі, яка має підтверджену відповідність шляхом створення комплексної системи захисту інформації (КСЗІ) з отриманням Атестата відповідності.

Специфіка використання бездротових мереж на державних об'єктах регулюється такими основними нормативними документами:

1. **Положення про технічний захист інформації в Україні**, затверджене Указом Президента України. Воно визначає загальні засади діяльності щодо запобігання витоку інформації технічними каналами.
2. **НД ТЗІ 2.5-010-03 «Вимоги до захисту інформації WEB-сторінок від несанкціонованого доступу»** та інші профільні стандарти Департаменту спеціального зв'язку та захисту інформації (ДССЗЗІ).
3. **Постанова Кабінету Міністрів України №1357**, яка визначає вимоги до захисту державних інформаційних ресурсів.

Важливою особливістю регулювання є те, що використання Wi-Fi мереж у державних установах часто обмежується або підлягає суворому контролю через складність гарантування цілісності та конфіденційності даних у відкритому радіоефірі. Згідно з вимогами ДССЗЗІ, бездротові мережі, що використовуються для передачі службової інформації, обов'язково повинні використовувати засоби криптографічного захисту інформації (ЗКЗІ), що мають сертифікат відповідності в Україні.

Особливе місце в нормативному регулюванні займає контроль за «чистотою ефіру». Посадові особи, відповідальні за ІТ-безпеку на держоб’єктах, зобов’язані проводити періодичний моніторинг радіочастотного спектра з метою виявлення:

- несанкціоновано встановлених точок доступу («Rogue Access Points»);
- сторонніх пристроїв, що намагаються отримати доступ до внутрішніх ресурсів;
- джерел випромінювання, що можуть спричиняти витік інформації за межі контрольованої зони.

Таким чином, розроблення програмного засобу для пасивного сканування бездротових мереж повністю відповідає вимогам нормативних документів щодо регулярного аудиту безпеки. Пасивний метод сканування, обраний у даній роботі, є найбільш прийнятним з точки зору нормативного регулювання, оскільки він не створює активних завад у радіоефірі та не потребує додаткових дозволів на випромінювання, при цьому забезпечуючи виконання вимог щодо контролю за станом технічного захисту об'єкта.

Нормативно-правова база також вимагає документування результатів кожної перевірки. Це обґрунтовує необхідність реалізації в програмі модуля генерації звітів, що міститимуть детальні технічні параметри виявлених мереж, які згодом можуть бути використані для формування актів перевірки стану ТЗІ в установі.

1.2. Огляд існуючих програмних засобів моніторингу

Для забезпечення ефективного контролю за станом безпеки бездротових мереж на об’єктах державної власності необхідно використовувати спеціалізоване програмне забезпечення. Сучасний ринок пропонує широкий спектр рішень, які можна класифікувати за типом ліцензії, режимом роботи (активний чи пасивний) та функціональним призначенням. Розглянемо детальніше найбільш розповсюджені інструменти, що використовуються фахівцями з технічного захисту інформації.

Kismet (Wireless Network Detector, Sniffer and IDS). Kismet є потужним інструментом з відкритим вихідним кодом, який працює на рівні канального шару моделі OSI. Його головна особливість полягає у виключно пасивному зборі даних. Програма переводить мережеву карту в режим моніторингу (monitor mode), що дозволяє «прослуховувати» всі пакети в ефірі, не надсилаючи жодного байта у відповідь.

- *Технічні можливості:* виявлення мереж за протоколами 802.11a/b/g/n/ac, ідентифікація прихованих точок доступу (hidden SSID) шляхом аналізу трафіку асоціації клієнтів, детектування типових атак на Wi-Fi.
- *Обмеження:* незважаючи на високу ефективність, Kismet має складний текстовий або веб-інтерфейс, що потребує специфічних знань Linux-систем. Крім того, робота з ним у середовищі Windows є вкрай нестабільною та обмеженою.

Acrylic Wi-Fi Professional. Це комерційне рішення, орієнтоване на глибокий аналіз пакетів у реальному часі. Програма дозволяє отримувати інформацію про швидкість передачі даних, типи шифрування та навіть перелік підключених до точок доступу клієнтів.

- *Технічні можливості:* підтримка моніторингу на частотах 2.4 ГГц та 5 ГГц, виведення детальних графіків завантаженості каналів, ідентифікація виробників обладнання за базою OUI.

- *Обмеження:* закритість коду є критичним фактором для державних установ, оскільки неможливо гарантувати відсутність «закладок» у самому ПЗ. Також вартість професійної ліцензії створює фінансове навантаження при масштабуванні на велику кількість об'єктів.

NetSpot (Survey and Wireless Analysis). Даний засіб позиціонується як інструмент для радіопланування. Він дозволяє створювати візуальні «теплові карти» (heatmaps) покриття об’єкта.

- *Технічні можливості:* аналіз співвідношення сигнал/шум (SNR), візуалізація зон інтерференції, автоматичне формування звітів про якість сигналу.
- *Обмеження:* основний акцент розробників зроблено на продуктивності мережі, а не на її безпеці. Програма слабо підходить для виявлення специфічних вразливостей, таких як слабкі алгоритми хешування або некоректні налаштування аутентифікації.

Wireshark (Network Protocol Analyzer). Це де-факто стандарт у світі аналізу мережевих протоколів. За наявності спеціалізованого адаптера (наприклад, AirPcap) Wireshark дозволяє розкладати кожен Wi-Fi фрейм на окремі поля.

- *Технічні можливості:* фільтрація трафіку за будь-яким критерієм, можливість розшифрування WPA2 трафіку при наявності ключа, експорт даних у безліч форматів.
- *Обмеження:* інструмент призначений для точкової діагностики, а не для постійного панорамного моніторингу ефіру. Автоматизована оцінка рівня безпеки за шкалою «небезпечно/надійно» у Wireshark відсутня.

Для наочності проведемо порівняльний аналіз розглянутих засобів за ключовими критеріями, що є важливими для розгортання в державних установах (табл. 1.1).

Таблиця 1.1

Порівняльний аналіз програмних засобів моніторингу бездротових мереж

Критерій порівняння	**Kismet**	**Acrylic Wi-Fi**	**NetSpot**	**Розроблений проєкт**
Режим сканування	Пасивний	Активний/ Пасивний	Активний	**Пасивний**
Кросплат-формність	Обмежена	Лише Windows	Win / macOS	**Повна (Win/ Lin/mac)**
Складність інтерфейсу	Висока	Середня	Низька	**Низька**
Оцінка рівня безпеки	Базова	Професійна	Відсутня	**Автоматизована**
Генерація PDF-звітів	Ні (тільки логи)	Так (платна)	Так	**Так (інтегрована)**
Тип ліцензії	Open Source	Комерційна	Комерцій-на	**Власна розробка**

Джерело: складено автором за матеріалами.

Аналіз таблиці 1.1 демонструє, що існуючі рішення або мають занадто високий поріг входження (Kismet, Wireshark), або є комерційними продуктами із закритим кодом. Це підтверджує необхідність розробки системи, яка поєднувала б у собі кросплатформність, пасивний метод збору даних та автоматизовану систему оцінки ризиків з можливістю швидкої генерації офіційної звітності.

1.3. Порівняльний аналіз аналогів та обґрунтування розробки власного рішення

Проведений у попередньому підрозділі огляд продемонстрував, що ринок програмного забезпечення для моніторингу бездротових мереж є досить насиченим, проте більшість рішень орієнтовані або на вузькоспеціалізовані наукові дослідження, або на комерційне використання в бізнес-секторі. Для державних об'єктів України існуючі аналоги мають низку критичних недоліків, які можна систематизувати за трьома напрямами:

1. **Проблема довіри та безпеки (Security Trust).** Використання комерційного ПЗ із закритим кодом (Acrylic, NetSpot) у державних установах створює ризики наявності недекларованих можливостей. Для об'єктів критичної інфраструктури пріоритетним є використання власних розробок або систем із повністю контрольованим вихідним кодом.
2. **Технічна складність та кросплатформність.** Більшість ефективних безкоштовних засобів (Kismet) орієнтовані на ОС Linux і потребують глибоких знань командного рядка. У той же час, робочі станції адміністраторів у держустановах часто працюють під управлінням ОС Windows або macOS.
3. **Відсутність інтегрованої звітності за вітчизняними стандартами.** Існуючі засоби не мають вбудованої логіки для автоматичного формування актів перевірки чи звітів українською мовою з фіксацією специфічних параметрів безпеки.

Нижче наведено порівняльну діаграму (Рис. 1.1) концептуальної моделі вибору між аналогами та власною розробкою, що підкреслює переваги проєктованої системи.

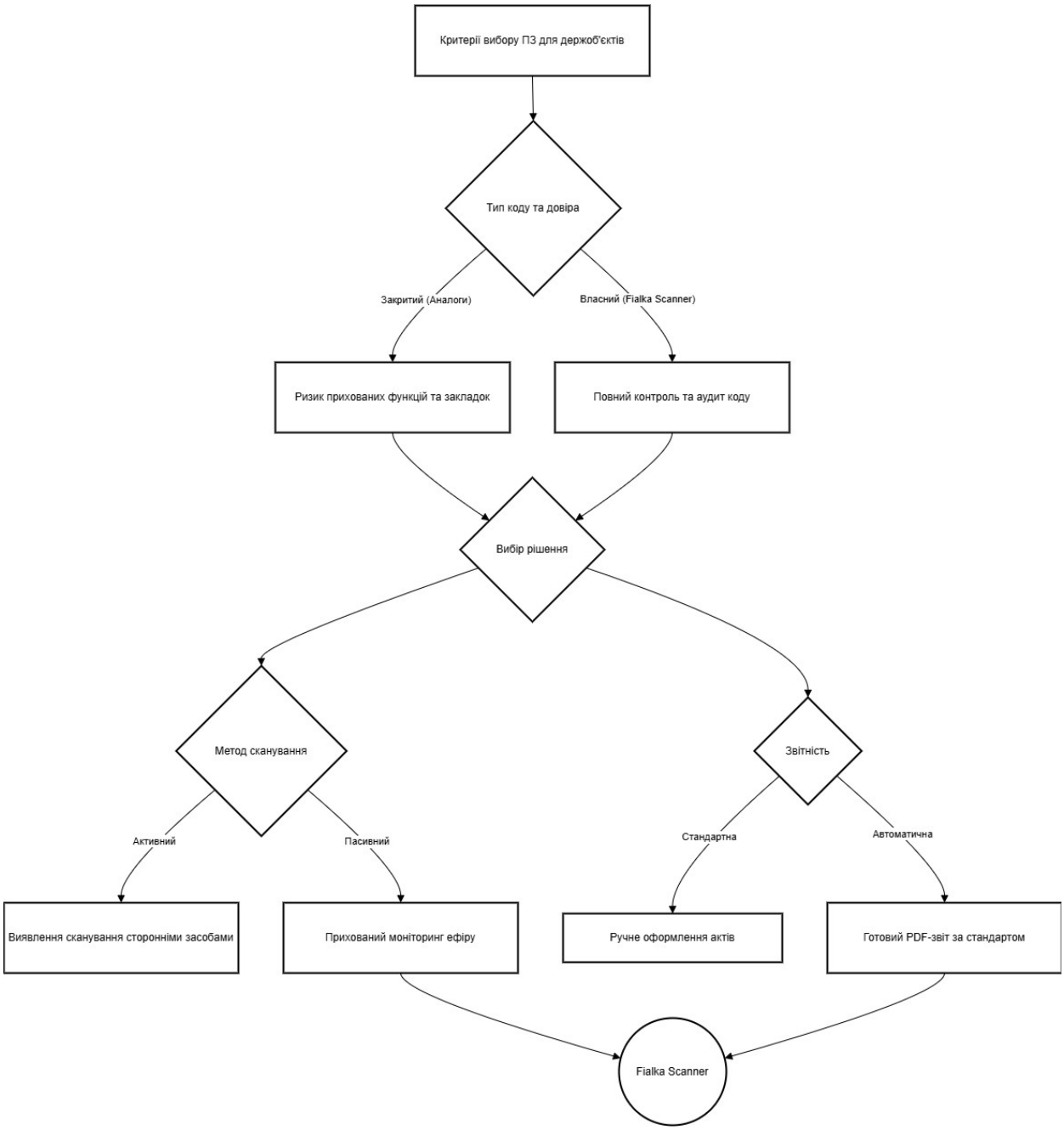

Рис. 1.1 Порівняльна діаграма концептуальної моделі вибору між аналогами та власною розробкою.

Обґрунтування розробки власного програмного продукту базується на впровадженні наступних унікальних функцій, які відсутні в комплексі у розглянутих аналогів:

- **Інтегральний показник безпеки:** замість відображення лише технічних параметрів (RSSI, канал), система автоматично обчислює рівень загрози (критичний, середній, надійний) на основі алгоритму,

що враховує тип шифрування, потужність сигналу та наявність SSID у «білому списку».

- **Кросплатформний рівень абстракції:** використання мови Python та обгорток над системними утилітами (`netsh`, `nmcli`, `airport`) дозволяє програмі працювати в будь-якій ОС без зміни коду.
- **Локальна база даних (SQLite):** на відміну від більшості сканерів, що працюють «тут і зараз», наша програма зберігає історію всіх спостережень, що дозволяє виявляти появу нових точок доступу протягом тривалого часу (ретроспективний аналіз).
- **Модуль звітності:** генерація PDF-документів з логотипом організації, датою та електронним підписом значно спрощує роботу офіцера безпеки.

Таким чином, розроблення власного рішення є не лише технічно доцільним, а й стратегічно обґрунтованим кроком для забезпечення належного рівня технічного захисту інформації на державних об'єктах.

РОЗДІЛ 2

ПРОЄКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ МОНІТОРИНГУ

2.1. Визначення функціональних та технічних вимог до системи

На етапі проєктування системи було визначено перелік вимог, що забезпечують ефективне виконання завдань з моніторингу бездротових мереж на об'єктах державної власності. Вимоги поділяються на функціональні (що система повинна робити) та технічні (яким чином і на якому обладнанні вона має функціонувати).

Функціональні вимоги:

1. **Пасивний збір даних:** програма повинна виявляти доступні Wi-Fi мережі, зчитуючи інформацію з ефіру без відправки активних запитів (probe requests), що забезпечує прихованість роботи.
2. **Ідентифікація технічних параметрів:** для кожної знайденої точки доступу система має визначати:
    - унікальний ідентифікатор (BSSID) та назву мережі (SSID);
    - робочий канал та відповідний частотний діапазон (2.4 ГГц, 5 ГГц або 6 ГГц);
    - потужність сигналу в децибелах (RSSI) або відсотках;
    - тип використовуваного шифрування та метод автентифікації.
3. **Автоматизована оцінка безпеки:** система повинна самостійно класифікувати рівень загрози кожної мережі за шкалою від «критичного» до «надійного» на основі виявлених вразливостей (наприклад, відкриті мережі або застарілий протокол WEP).
4. **Збереження історії:** реалізація локального сховища для ведення логів сканування, що дозволяє відстежувати появу нових пристроїв у часі.

5. **Генерація звітності:** формування підсумкового PDF-документа, що містить аналітичну інформацію про стан ефіру на об'єкті, включаючи графіки та таблиці.
6. **Візуалізація даних:** відображення результатів у зручному графічному інтерфейсі з можливістю фільтрації та детального перегляду кожної мережі.

Технічні вимоги:

1. **Кросплатформність:** забезпечення коректної роботи в операційних системах сімейства Windows (через утиліту `netsh`), Linux (через `nmcli`) та macOS (через утиліту `airport`).
2. **Мова розробки:** Python 3.10+ як основний інструмент реалізації логіки та взаємодії з ОС.
3. **Графічний інтерфейс:** використання фреймворку PyQt6 для створення сучасного та швидкого UI з підтримкою української локалізації.
4. **База даних:** застосування вбудованої СУБД SQLite для забезпечення автономності програми (не потребує встановлення окремого сервера БД).
5. **Вимоги до апаратного забезпечення:** наявність стандартного бездротового адаптера з підтримкою режимів, необхідних для зчитування інформації про навколишні мережі.
6. **Автономність:** програма повинна збиратися в єдиний виконуваний файл (.exe для Windows) за допомогою PyInstaller для спрощення розгортання на робочих місцях без встановленого інтерпретатора Python.

Окрему увагу приділено етичним принципам та обмеженням: програмний засіб не містить функцій для проведення атак (брутфорс, деавтентифікація), що робить його використання легітимним у межах правового поля України щодо аудиту власних інформаційних систем.

Візуалізація того, як модулі взаємодіють між собою (Рис. 2.1)

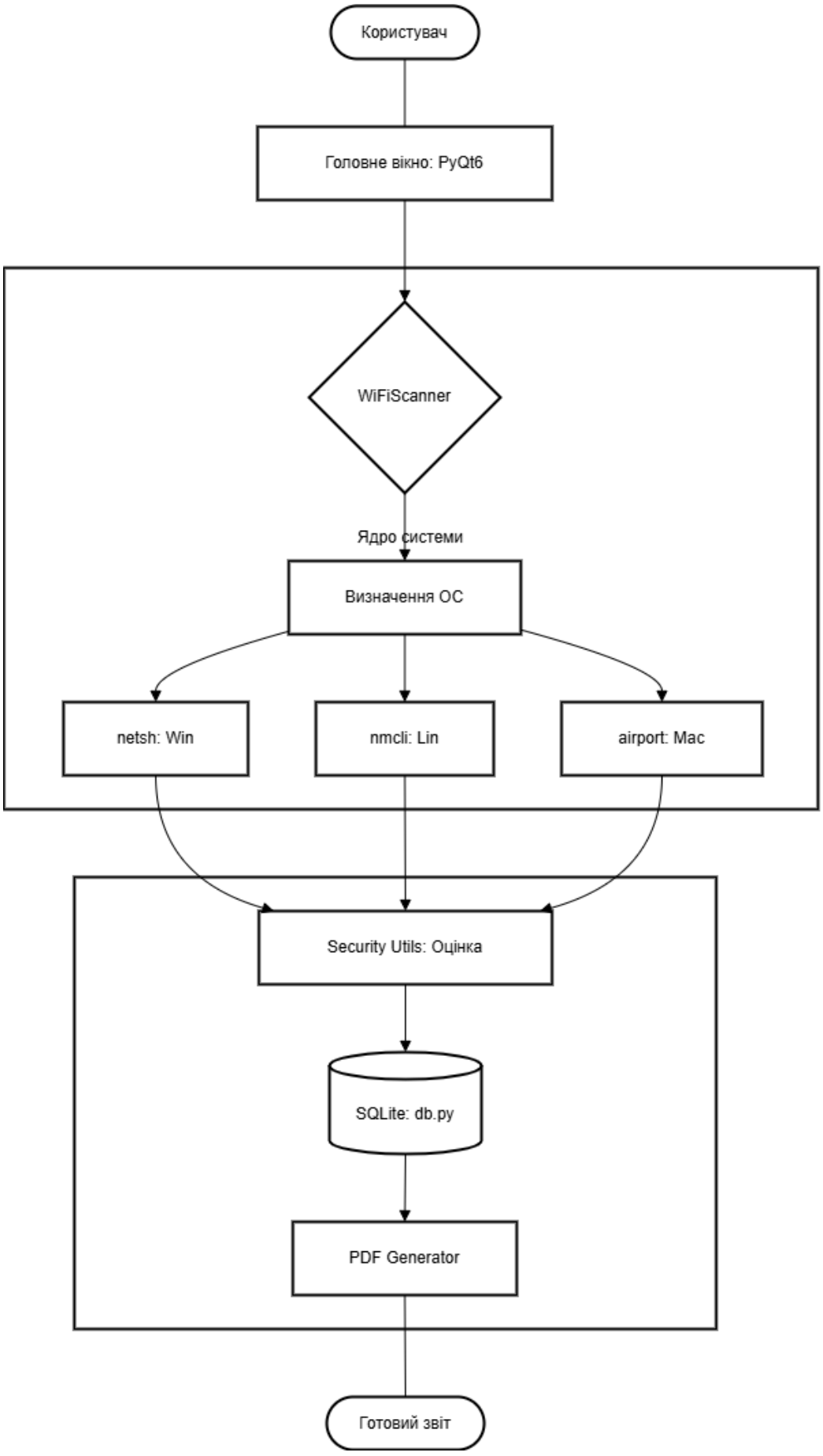

Рис. 2.1 Схема функціональної структури ПЗ

2.2. Обґрунтування вибору архітектури та технологічного стеку

При розробленні автоматизованої системи вибір технологічного стеку був зумовлений необхідністю поєднання високої швидкості розробки, кросплатформності та забезпечення максимального рівня довіри до програмного коду, що є критичним для об'єктів державної власності.

Вибір архітектурного підходу. Програмний комплекс побудований за принципами модульної архітектури. Це дозволяє відокремити логіку збору даних (Backend) від графічного інтерфейсу (Frontend), що забезпечує стабільність роботи системи. Основні компоненти системи розділені на логічні рівні:

- *Рівень взаємодії з ОС:* реалізований через модуль `WiFiScanner`, який динамічно адаптується до типу операційної системи.
- *Рівень бізнес-логіки:* містить алгоритми аналізу та оцінки безпеки (`security.py`).
- *Рівень даних:* забезпечує локальне зберігання та генерацію звітів.

При виборі конкретних програмних рішень для реалізації кожного рівня особлива увага приділялася мінімізації сторонніх залежностей та забезпеченню автономності роботи ПЗ без необхідності постійного підключення до мережі Інтернет. Це критично важливо для державних об'єктів, де доступ до зовнішніх ресурсів може бути обмежений вимогами безпеки.

Основними критеріями порівняння при формуванні технологічного стеку стали: швидкість обробки мережевих пакетів, наявність відкритих бібліотек для криптографічного аналізу, простота портування коду між різними ОС та можливість створення єдиного виконуваного файлу для спрощеного розгортання на робочих станціях.

Для наочного обґрунтування прийнятих рішень проведено порівняльний аналіз обраних технологій із найбільш розповсюдженими альтернативами, результати якого наведено в таблиці 2.1.

Таблиця 2.1

Обґрунтування вибору програмних інструментів

Компонент системи	**Обрана технологія**	**Переваги для державних об'єктів**	**Альтернатива**
Мова розробки	**Python 3.10+**	Висока читабельність коду для аудиту, величезна кількість бібліотек безпеки.	C++, Java
Інтерфейс (GUI)	**PyQt6**	Висока продуктивність, підтримка сучасних стандартів Windows 10/11.	Electron, Tkinter
База даних	**SQLite**	Повна автономність (файл), дані не передаються по мережі.	MySQL, PostgreSQL
Звітність	**ReportLab**	Пряма генерація PDF без стороннього ПЗ, контроль верстки.	MS Word API

Джерело: складено автором за матеріалами.

Обґрунтування вибору мови програмування Python. Вибір мови Python 3.10+ обумовлений її роллю як «мови-клею» (glue language), що дозволяє ефективно взаємодіяти з системними процесами через модуль `subprocess`. Це критично для виклику низькорівневих утиліт сканування. Крім того, кросплатформність Python дозволяє використовувати програму на різних типах автоматизованих робочих місць (АРМ) без зміни програмного ядра.

Графічний фреймворк PyQt6. На відміну від веб-рішень (наприклад, Electron), PyQt6 створює справжній нативний додаток. Це важливо для закритих систем, де використання браузерних технологій може створити додаткові вектори атак. Бібліотека дозволяє інтегрувати складні графіки `matplotlib` безпосередньо у вікна програми.

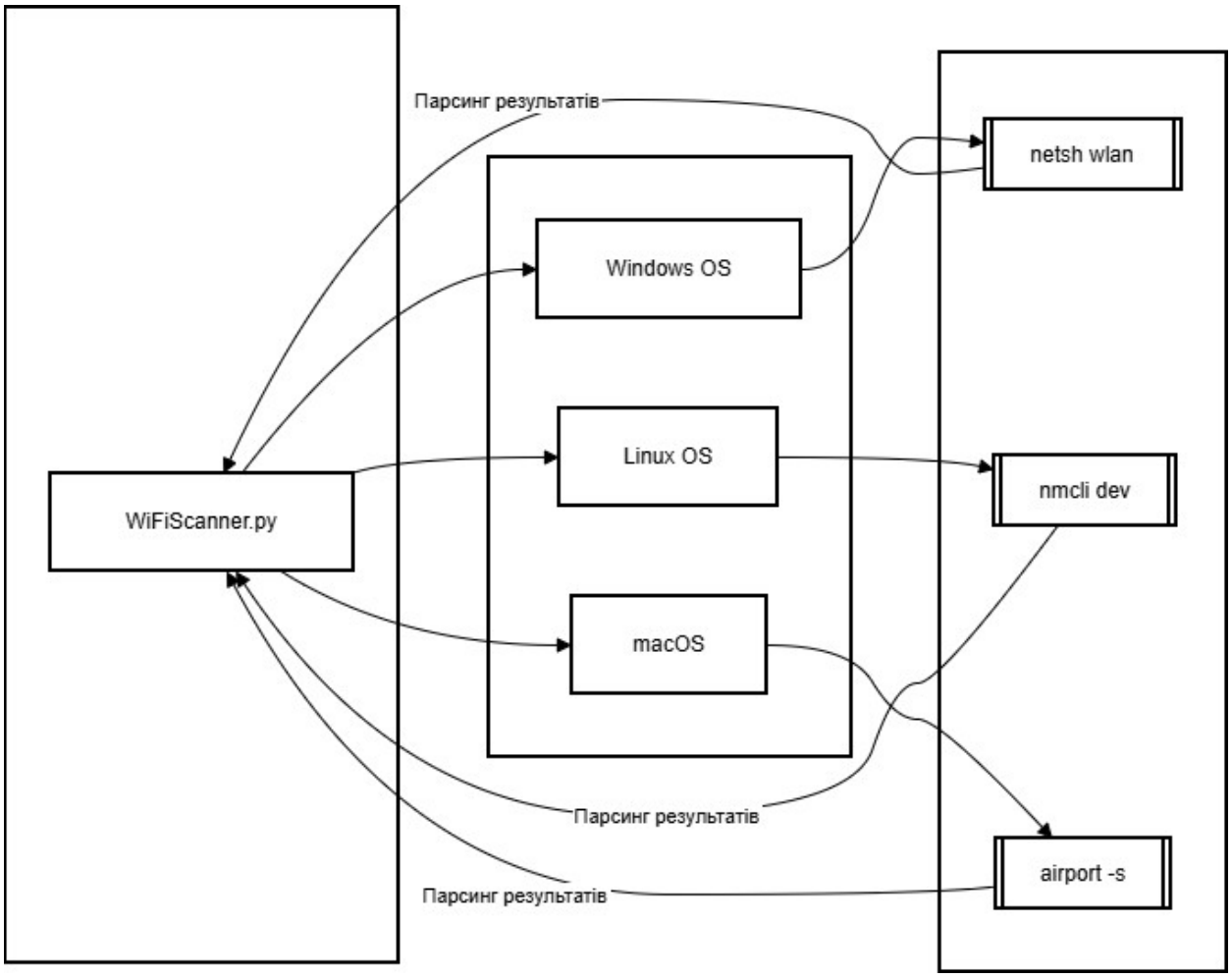

Рис. 2.2. Схема взаємодії модулів системи

Система управління базами даних SQLite. Використання SQLite є найбільш виправданим з точки зору ТЗІ. Оскільки база даних є звичайним файлом, адміністратор безпеки може легко контролювати доступ до неї, робити резервні копії або фізично знищувати дані за потреби. Відсутність активного мережевого порту БД унеможливлює дистанційні атаки на сховище історії сканувань.

Генерація аналітичної звітності. Бібліотека ReportLab вибрана для автоматизації створення технічних звітів. Це дозволяє перетворювати результати пасивного моніторингу на офіційні документи, що містять таблиці вразливостей та графічний аналіз ефіру.

2.3. Розробка алгоритму пасивного сканування та кросплатформної взаємодії з ОС

Основною функціональною особливістю системи є її здатність здійснювати моніторинг бездротового ефіру в пасивному режимі на різних операційних системах без зміни програмного коду. Це реалізовано за допомогою розробленого класу `WiFiScanner`, який виконує роль абстрактного рівня між логікою програми та системними утилітами ОС.

Алгоритм роботи модуля сканування. Процес збору даних починається з ідентифікації поточної операційної системи за допомогою бібліотеки `platform`. Залежно від отриманого результату, програма обирає один із трьох сценаріїв взаємодії:

1. **Windows-сценарій:** використовується системна утиліта `netsh wlan show networks mode=bssid`. Програма ініціює виклик команди через модуль `subprocess`, після чого виконує складний парсинг текстового виводу. Оскільки Windows може кешувати результати, в алгоритмі передбачена затримка (`time.sleep`) та повторний запит у разі виявлення лише однієї мережі, що підвищує актуальність даних.
2. **Linux-сценарій:** взаємодія відбувається через інтерфейс командного рядка `nmcli` (Network Manager CLI). Використання прапорців `-t` (terse) та `-f` (fields) дозволяє отримати дані у структурованому вигляді, що значно прискорює процес обробки та знижує навантаження на систему.
3. **macOS-сценарій:** використовується низькорівнева утиліта `airport`. Алгоритм перераховує отримані значення потужності сигналу (RSSI) у відсоткову шкалу для уніфікації відображення в інтерфейсі.

Для візуалізації логіки роботи розроблено алгоритмічну схему функціонування модуля (рис. 2.4).

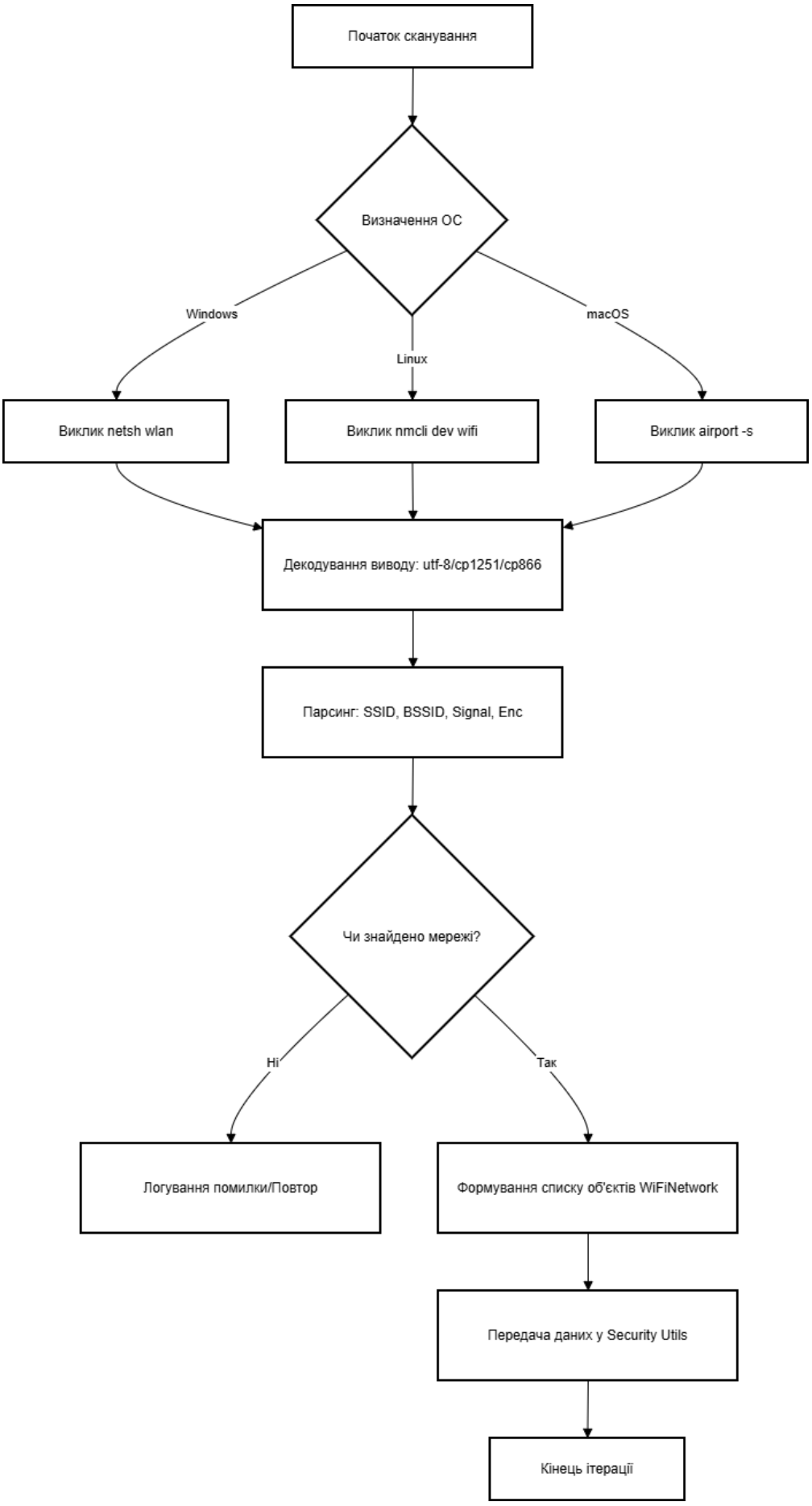

Рис. 2.4 Алгоритмічна схема функціонування модуля

Механізм обробки кодувань. Важливою технічною деталлю при розробці була реалізація методу `_decode_output`. У державних установах часто використовуються локалізовані версії ОС (особливо Windows), де вивід консольних утиліт може бути в кодуваннях `cp1251` або `cp866`. Розроблений алгоритм автоматично перебирає найбільш імовірні кодування, що гарантує коректне відображення кириличних назв мереж (SSID) та технічних термінів у звітах.

Пасивний метод збору даних. На відміну від активних сканерів, які розсилають «Probe Request» пакети, розроблений алгоритм базується на зчитуванні інформації, яку точки доступу транслюють самостійно (Beacon frames). Це дозволяє:

- уникати зайвого радіовипромінювання на об'єкті;
- залишатися непомітним для систем виявлення вторгнень (WIDS) сторонніх мереж;
- проводити моніторинг без переривання поточного мережевого з’єднання користувача.

Результатом роботи алгоритму є набір уніфікованих об'єктів `WiFiNetwork`, які містять очищені та структуровані дані, готові для подальшого аналізу безпеки та збереження в базу даних SQLite.

2.4. Методика оцінки рівня безпеки мережі (визначення критеріїв ризику)

Для автоматизації процесу аудиту бездротових мереж у системі реалізовано методику бальної оцінки ризиків. Методика базується на аналізі технічних параметрів кожної виявленої точки доступу та порівнянні їх із встановленими стандартами безпеки для державних установ.

Критерії оцінки та вагові коефіцієнти. Оцінка кожної мережі формується на основі трьох ключових груп параметрів:

1. **Тип шифрування (Encryption Score):** найважливіший показник, що визначає стійкість мережі до зламу.
2. **Потужність сигналу (Proximity Risk):** визначає фізичну доступність мережі за межами контрольованої зони об'єкта.
3. **Ідентифікація (SSID Status):** перевірка на наявність мережі у «білому списку» дозволених пристроїв та виявлення прихованих ідентифікаторів.

Алгоритм обчислення інтегрального показника безпеки можна представити у вигляді таблиці штрафних балів (табл. 2.2)

Таблиця 2.1

Система оцінки ризиків бездротових мереж

Параметр мережі	**Значення**	**Рівень загрози**	**Дія системи**
Протокол захисту	Open / None	**Критичний**	Негайне сповіщення
Протокол захисту	WEP / WPA	**Високий**	Рекомендація з оновлення
Протокол захисту	WPA2 / WPA3	**Низький**	Захищено
Назва (SSID)		**Середній**	Потребує додаткової перевірки
Потужність сигналу	> 80% (дуже висока)	**Середній**	Ризик витоку за межі будівлі

Джерело: складено автором за матеріалами.

Класифікація рівнів безпеки. На основі сукупності отриманих даних програма присвоює кожній мережі один із п'яти статусів, які візуалізуються в інтерфейсі відповідним кольором:

1. **Надійний (Safe):** Використовується WPA3 або WPA2 Enterprise, SSID відомий, сигнал у межах норми.

2. **Добрий (Good):** WPA2 Personal із сильним паролем, відсутність аномалій.
3. **Середній (Warning):** Приховані SSID або занадто потужний сигнал, що виходить за межі території.
4. **Небезпечний (Danger):** Використання застарілих протоколів (WEP/WPA), які вразливі до швидкого зламу.
5. **Критичний (Critical):** Повністю відкрита мережа (Open) без шифрування, що дозволяє перехоплювати трафік будь-якому користувачу.

Логіка прийняття рішення модулем `security.py` представлена на схемі (рис. 2.5)

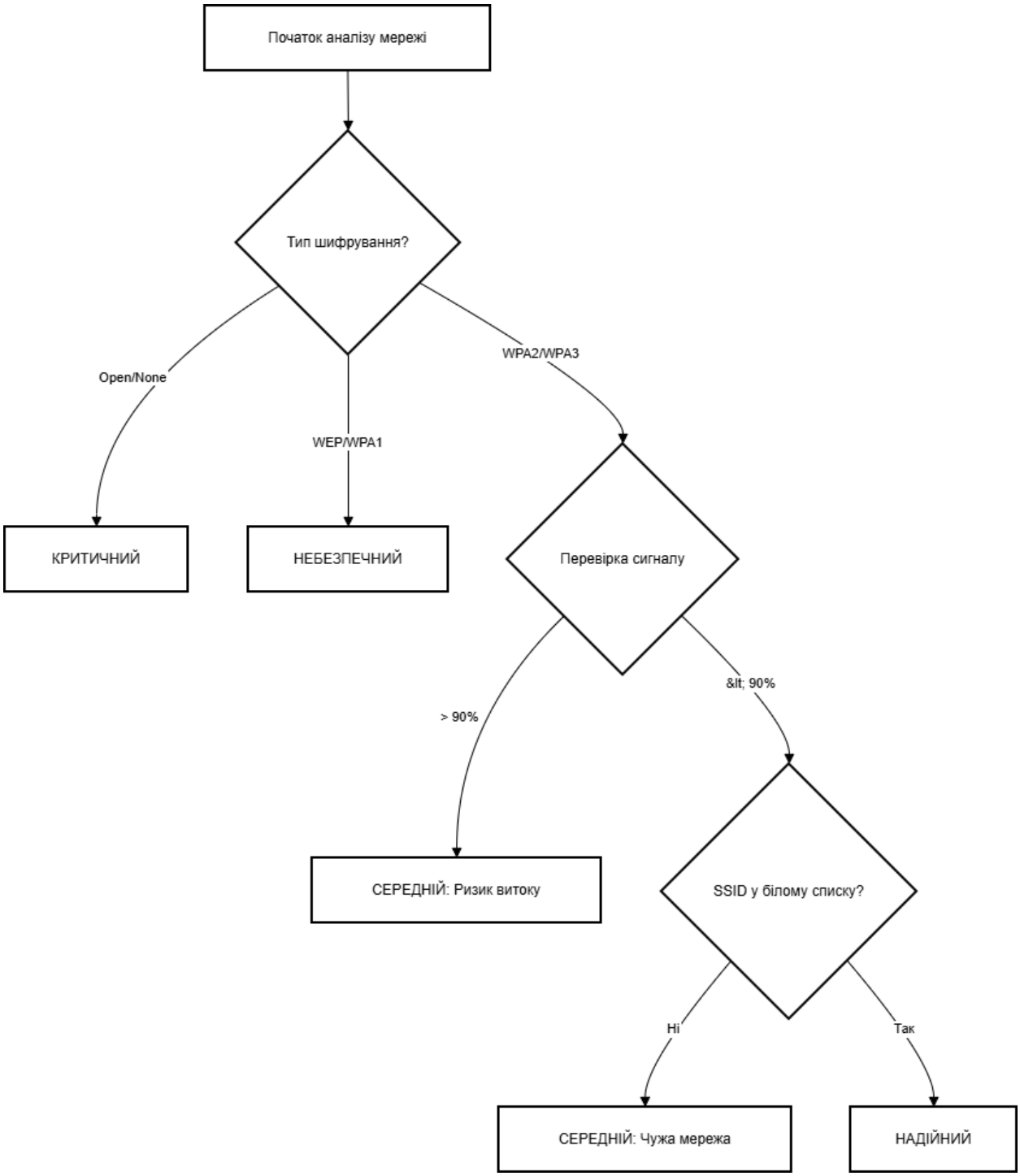

Рис. 2.5 Логіка прийняття рішення модулем `security.py`

Ця методика дозволяє автоматизувати роботу офіцера безпеки: замість аналізу десятків технічних параметрів він отримує готовий перелік об'єктів, що потребують негайної уваги. Результати цієї оцінки автоматично заносяться до звіту, що забезпечує об’єктивність проведення аудиту.

РОЗДІЛ 3

ПРОГРАМНА РЕАЛІЗАЦІЯ СКАНЕРА БЕЗДРОТОВИХ МЕРЕЖ

3.1. Структура програмного комплексу та взаємодія модулів

Реалізація сканера базується на модульній структурі, що відповідає сучасним стандартам розробки складних програмних систем. Такий підхід дозволяє ізолювати окремі функціональні блоки, що полегшує процес налагодження (debugging) та подальшої модернізації кожної підсистеми незалежно від інших.

Організація файлової структури. Вихідний код проєкту організований у логічні пакети (директорії), кожен з яких виконує чітко визначену роль. Фізичне розмежування компонентів у середовищі розробки дозволяє уникнути сильної зв'язності коду.

Проєкт має таку ієрархію каталогів:

- **`main.py`** -центральна точка входу, яка запускає графічну оболонку та логування.
- **пакет `scanner/`** (`wifi_scanner.py`) -кросплатформне ядро для пасивного збору даних з ефіру.
- **пакет `ui/`** (`main_window.py`, `login_dialog.py`, `details_dialog.py`, `settings_dialog.py`) -графічні форми та логіка обробки подій користувача на базі фреймворку PyQt6.
- **пакет `database/`** (`db.py`) -рівень взаємодії із СУБД SQLite для збереження історії моніторингу.
- **пакет `report/`** (`pdf_generator.py`) -модуль формування та верстки підсумкових PDF-звітів.
- **пакет `utils/`** -допоміжні модулі для аналізу ризиків безпеки (`security.py`) та логування подій (`logger.py`).

- **допоміжні папки `logs/` та `data/`** -призначені для збереження текстових журналів роботи та самого файлу локальної бази даних відповідно.

Програмна реалізація базових сутностей та логіки сканування. Для забезпечення чистоти архітектури дані бездротового ефіру типізуються за допомогою дата-класів. На рисунку 3.1 представлено архітектурну основу модуля `wifi_scanner.py`, яка описує структуру збереження параметрів мережі `WiFiNetwork` та базовий керуючий клас `WiFiScanner`.

```python
from __future__ import annotations
import platform
import locale
import time
import re
import subprocess
from dataclasses import dataclass
from typing import Iterable

@dataclass
class WiFiNetwork:
    ssid: str
    bssid: str
    channel: str
    signal: int | None
    encryption: str

class WiFiScanner:
    def scan(self, band: str = "auto") -> list[WiFiNetwork]:
        system = platform.system().lower()
        if system == "windows":
            return list(self._scan_windows())
        if system == "linux":
            return list(self._scan_linux(band))
        if system == "darwin":
            return list(self._scan_macos())
        raise RuntimeError("Непідтримувана операційна система.")
```

Рис. 3.1 Структура сутностей та архітектура модуля wifi_scanner.py

Метод `scan` динамічно визначає тип поточної операційної системи (Windows, Linux чи macOS) на базі стандартної бібліотеки `platform` і

перенаправляє потік виконання на відповідні приватні методи збору даних, забезпечуючи повну ізоляцію платформно-залежного коду.

Взаємодія модулів комплексу. Взаємодія між розмежованими пакетами побудована на базі подієво-орієнтованої моделі за допомогою механізму сигналів та слотів Qt. Головне вікно програми (`main_window.py`) виступає в ролі центрального координатора. Схематично взаємозв'язок між модулями під час виконання стандартного робочого циклу програми зображено на рис. 3.2.

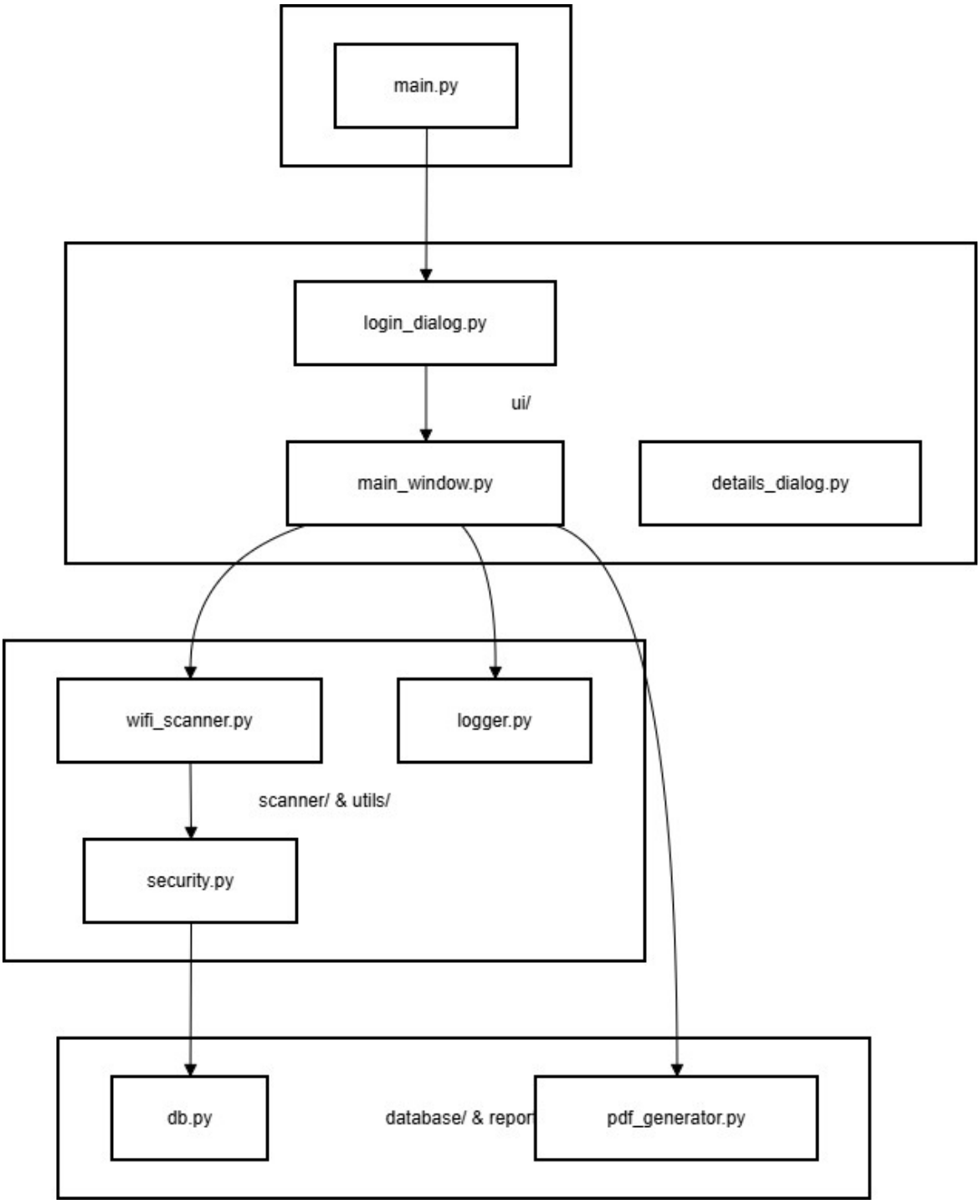

Рис. 3.2 Схема взаємозв’язку між модулями

Завдяки такій побудові, якщо на об'єкті державної власності змінюються вимоги до збереження інформації (наприклад, замість SQLite буде наказано використовувати іншу СКБД), розробнику достатньо модифікувати виключно вміст пакету `database/`, не змінюючи логіку роботи інтерфейсу чи кросплатформного сканера.

3.2. Розробка графічного інтерфейсу користувача (GUI).

Графічний інтерфейс користувача (GUI) автоматизованої системи розроблено з використанням сучасного фреймворку PyQt6. Основним завданням при проєктуванні інтерфейсу було забезпечення високої інформативності, швидкого відгуку при оновленні великих масивів даних та простоти сприйняття аналітичних показників оператором ТЗІ.

Головне вікно програми та вкладка «Сканування». Центральним елементом взаємодії користувача з комплексом є головне вікно, інтерфейс якого побудовано із застосуванням темної теми оформлення для зниження зорового навантаження на оператора. Верхня панель містить інформаційні лічильники: кількість критичних, підозрілих, загальних та зниклих з ефіру мереж.

Для відображення списку точок доступу інтегровано табличний віджет, який динамічно виводить параметри SSID, BSSID, номер каналу, рівень сигналу у відсотках, тип шифрування, рівень безпеки, пояснення, час останнього виявлення та поточний статус. Загальний вигляд вкладки «Сканування» представлено на рис. 3.3.

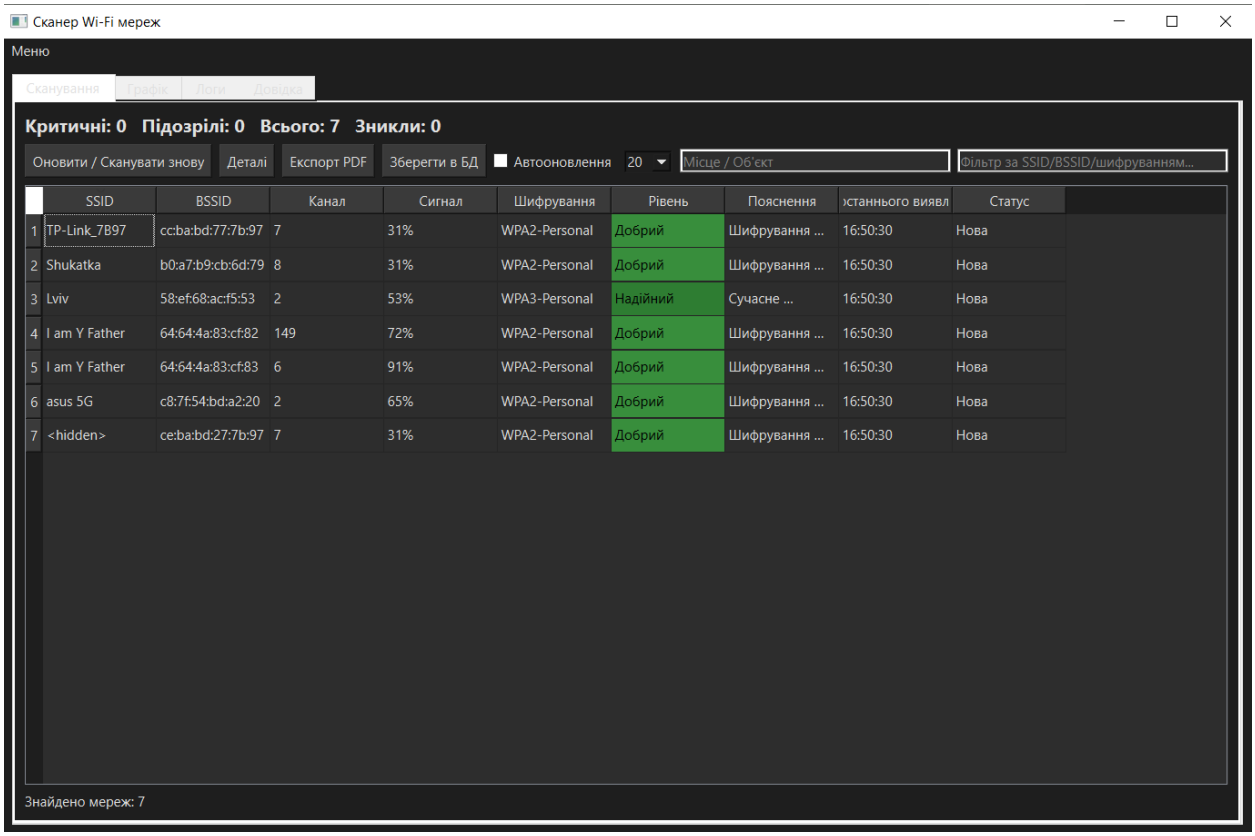

Рис. 3.3 Загальний вигляд вкладки «Сканування»

Колірна індикація ризиків реалізована безпосередньо у стовпці «Рівень»: безпечні мережі (наприклад, із сучасним протоколом WPA3) маркуються яскраво-зеленим кольором («Надійний»), тоді як потенційні загрози підсвічуються іншими відтінками для негайної фіксації оператором.

Картка детальної інформації про мережу. При виділенні конкретного запису та натисканні кнопки «Деталі» (або за допомогою подвійного кліку) ініціюється виклик діалогового вікна класу `DetailsDialog`. Воно виконує роль розширеної технічної карти точки доступу. Вигляд вікна детальної інформації наведено на рис. 3.4.

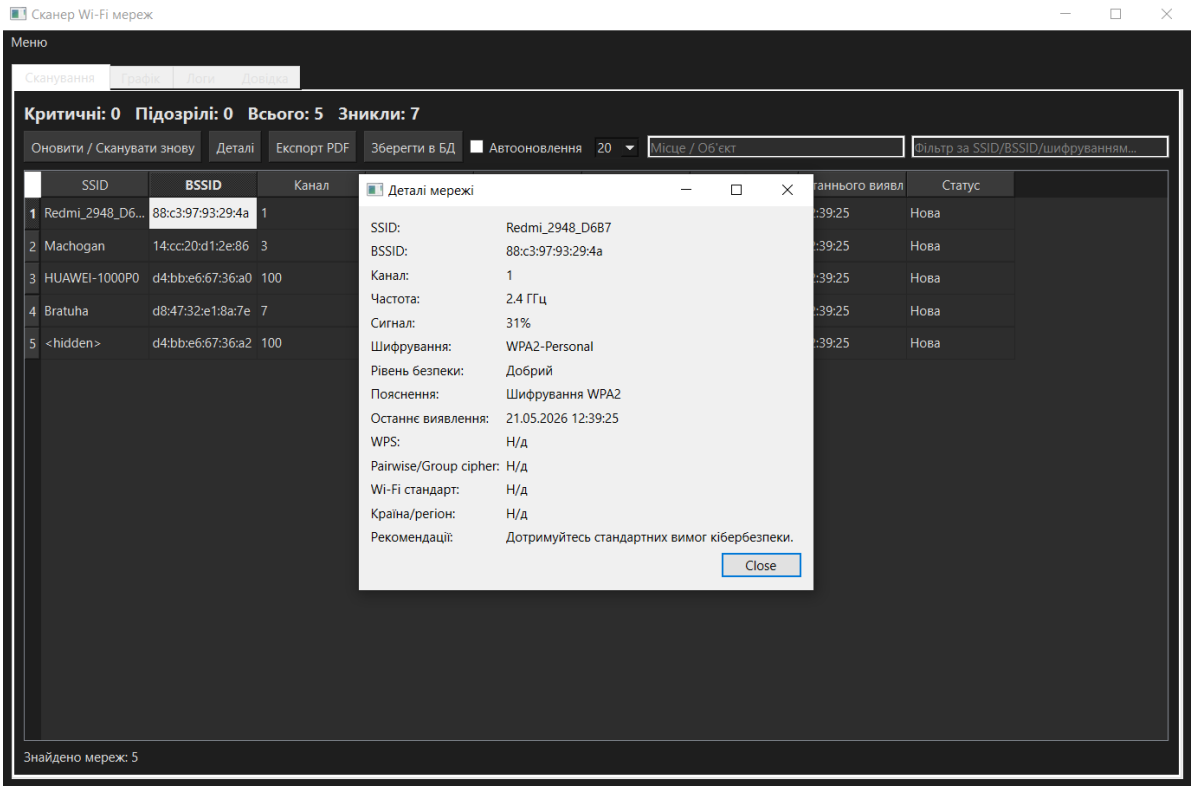

Рис. 3.4 Вигляд вікна детальної інформації

Цей інтерфейсний компонент дозволяє фахівцю ознайомитися з додатковими характеристиками, такими як точна частота роботи (наприклад, 2.4 ГГц), наявність технології WPS, типи попарних шифрів (Pairwise/Group cipher), країна/регіон та сформована системою автоматична рекомендація щодо поводження з цією мережею.

Модуль графічного аналізу ефіру. Вкладка «Графік» інтегрує можливості бібліотеки `matplotlib` для наочного представлення потужності сигналів топ-15 виявлених точок доступу. Побудована гістограма (рис. 3.5) дозволяє швидко оцінити фізичну наближеність джерел випромінювання до АРМ сканування.

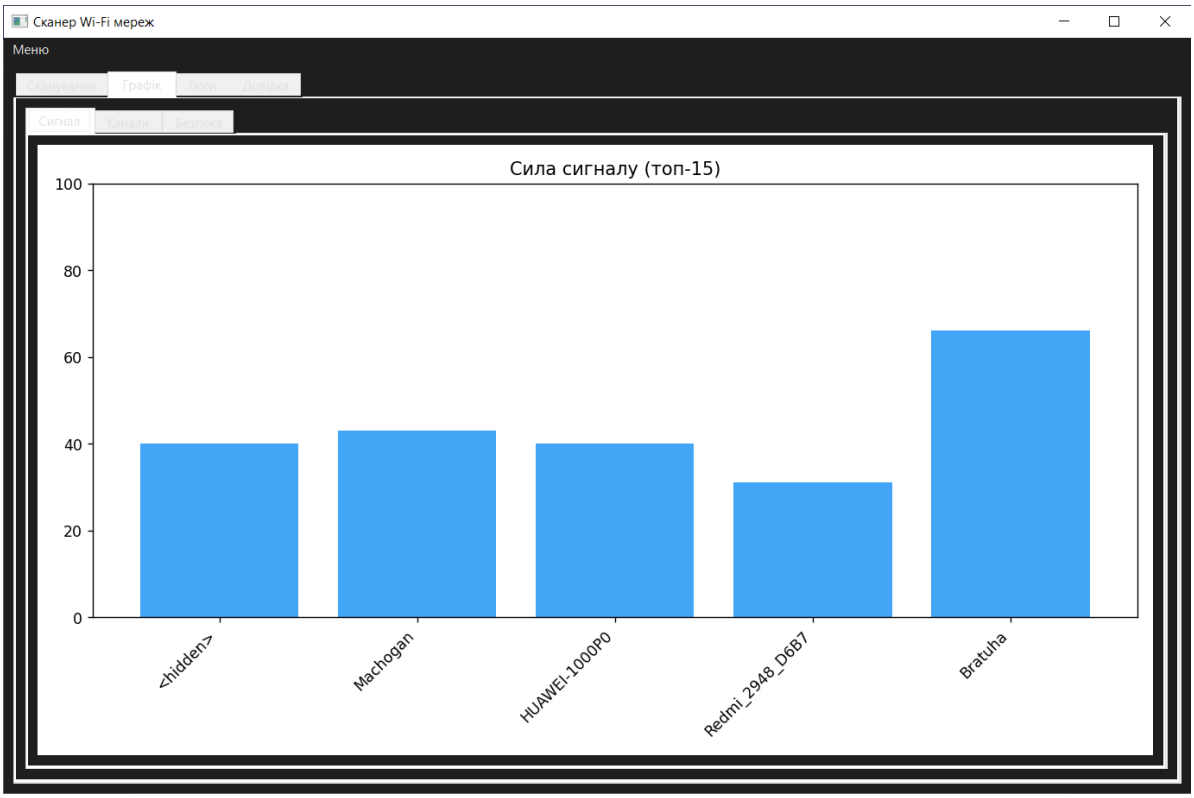

Рис. 3.5 Гістограма оцінки

Підсистема логування та довідкова інформація. Для забезпечення прозорості роботи програми та фіксації збоїв реалізовано вкладку «Логи» (рис. 3.6), куди в реальному часі виводяться відмітки часу та системні повідомлення про початок або завершення ітерацій сканування.

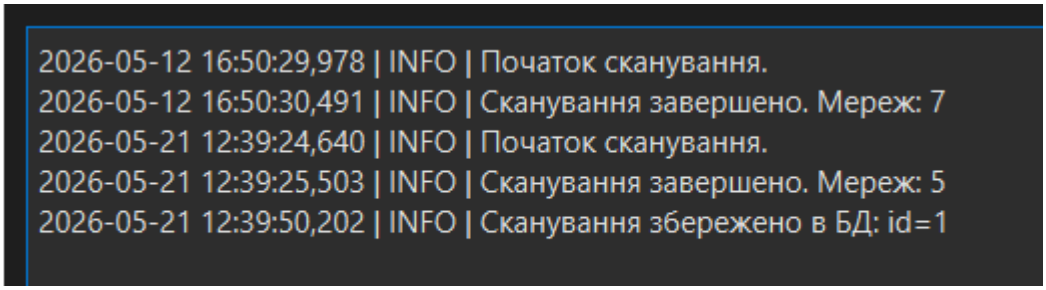

Рис. 3.6 Вкладка «Логи»

Вкладка «Довідка» (рис. 3.7) містить вбудовану експертну систему-підказку для оператора, де розписано градацію ризиків (Open, WEP, WPA, WPA2, WPA3) та базові рекомендації з кібербезпеки бездротового простору на об'єкті.

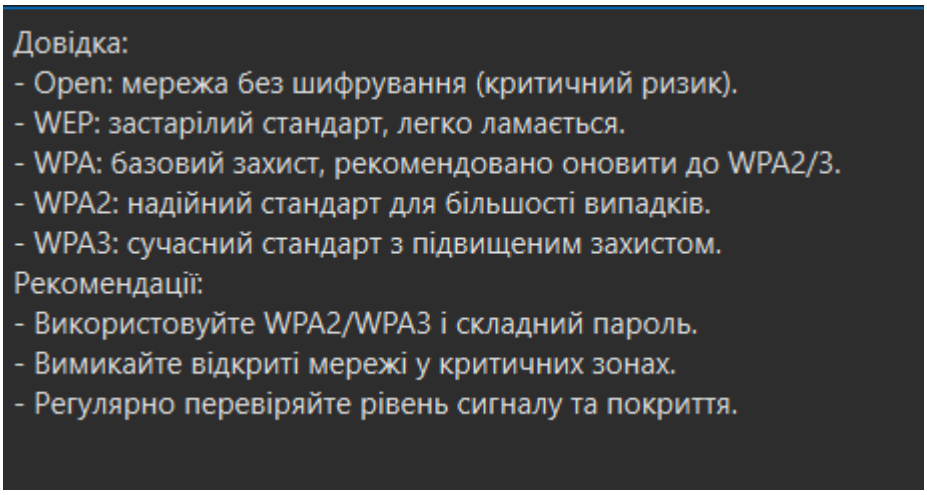

Рис. 3.7 Вкладка «Довідка»

На рисунку 3.8 наведено програмну реалізацію механізму перемикання вкладок та динамічного оновлення лічильників і статусів у головному вікні PyQt6.

```
from PyQt6.QtWidgets import QTableWidgetItem
from PyQt6.QtGui import QColor, QBrush

def update_network_table(self, networks: list[-Будь-який-Клас-]) -> None:
    self.tableWidget.setRowCount(0)
    for row_idx, net in enumerate(networks):
        self.tableWidget.insertRow(row_idx)

        # Заповнення текстових полів
        self.tableWidget.setItem(row_idx, 0, QTableWidgetItem(net.ssid))
        self.tableWidget.setItem(row_idx, 1, QTableWidgetItem(net.bssid))

        # Реалізація динамічного підсвічування клітинок
        rating_item = QTableWidgetItem(net.security_rating)
        if net.security_rating == "Надійний":
            rating_item.setBackground(QBrush(QColor("#4CAF50"))) # Зелений
            rating_item.setForeground(QBrush(QColor("#FFFFFF")))
        elif net.security_rating == "Добрий":
            rating_item.setBackground(QBrush(QColor("#2E7D32"))) # Темно-
зелений
            rating_item.setForeground(QBrush(QColor("#FFFFFF")))
        elif net.security_rating in ["Небезпечний", "Критичний"]:
            rating_item.setBackground(QBrush(QColor("#D32F2F"))) # Червоний
            rating_item.setForeground(QBrush(QColor("#FFFFFF")))

        self.tableWidget.setItem(row_idx, 5, rating_item)
```

Рис. 3.8 Фрагмент обробки подій та колірного кодування рівнів безпеки в main_window.py

Таким чином, розроблена архітектура GUI повністю забезпечує оператора інструментами для візуального контролю, детальної діагностики загроз та оперативного реагування на інциденти безпеки.

3.3. Реалізація підсистеми зберігання історії та генерації PDF-звітів

Для забезпечення ретроспективного аналізу безпеки бездротового простору та формування офіційної документації на об'єктах державної власності, в систему інтегровано дві взаємопов'язані підсистеми: довгострокового зберігання історії сканування та генерації аналітичних звітів.

Реалізація підсистеми зберігання історії (SQLite). Взаємодія з базою даних SQLite реалізована через модуль `db.py` (пакет `database/`). Оскільки архітектура системи передбачає повну локальність та автономність, база даних автоматично створюється у вигляді файлу `fialka_history.db` у робочій директорії програми при першому запуску.

При натисканні користувачем кнопки «Зберегти в БД» на головній панелі, поточний зріз ефіру з таблиці трансформується в серію SQL-запитів. Для оптимізації швидкодії та запобігання блокуванню графічного інтерфейсу, операції запису виконуються через механізм транзакцій. Процес ініціалізації збереження аналітичного звіту наведено на рис. 3.9.

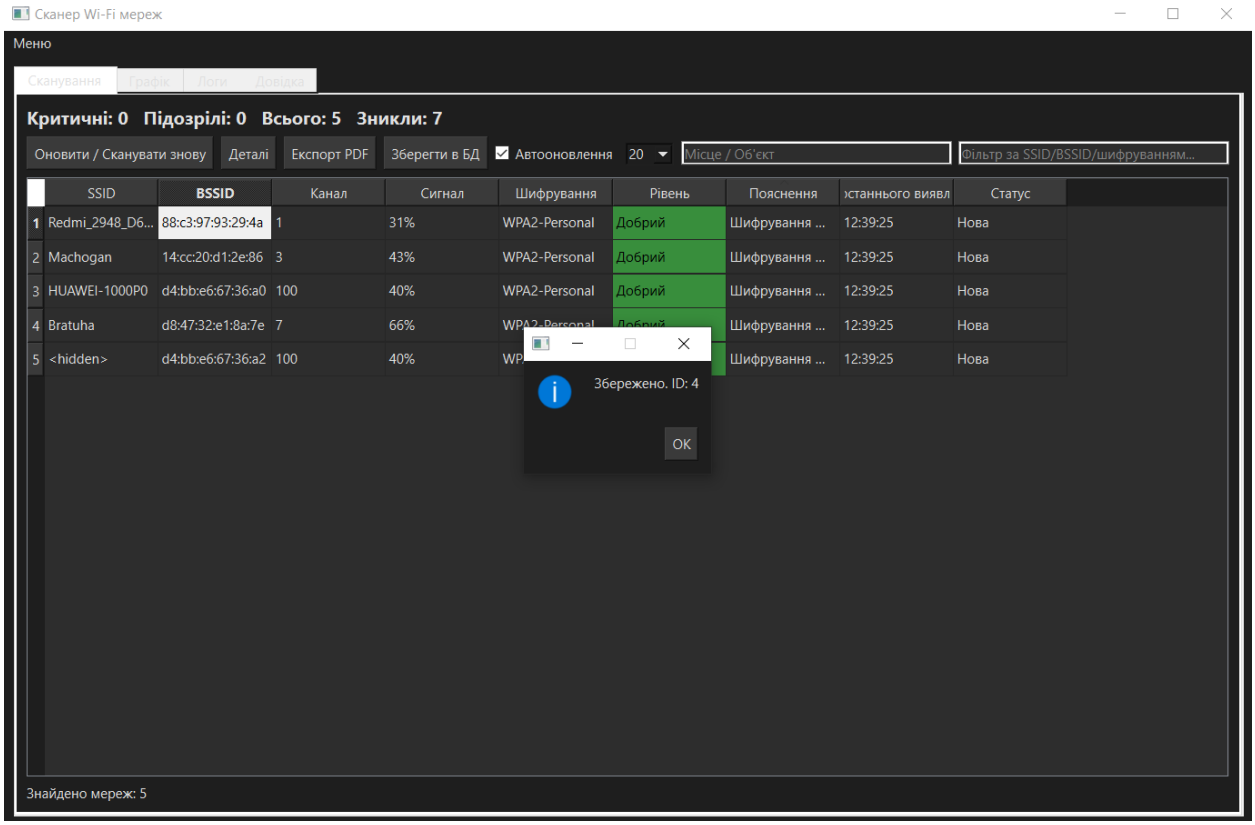

Рис. 3.9 Процес ініціалізації збереження аналітичного звіту

На рисунку 3.10 наведено програмний код ініціалізації таблиці результатів моніторингу та метод масового збереження виявлених мереж.

```python
import sqlite3
from datetime import datetime

class DatabaseManager:
    def __init__(self, db_path: str = "data/fialka_history.db") -> None:
        self.db_path = db_path
        self.init_db()

    def init_db(self) -> None:
        """Створення таблиці для збереження результатів сканування"""
        with sqlite3.connect(self.db_path) as conn:
            cursor = conn.cursor()
            cursor.execute("""
                CREATE TABLE IF NOT EXISTS scan_results (
                    id INTEGER PRIMARY KEY AUTOCTINCREMENT,
                    timestamp TEXT,
                    ssid TEXT,
                    bssid TEXT,
                    channel TEXT,
                    signal INTEGER,
                    encryption TEXT,
                    security_rating TEXT
                )
            """)
            conn.commit()

    def save_scan_results(self, networks: list) -> None:
        """Масове збереження отриманих мереж в одну транзакцію"""
        current_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        with sqlite3.connect(self.db_path) as conn:
            cursor = conn.cursor()
            # Використання executemany для максимальної швидкодії
            data_to_insert = [
                (current_time, net.ssid, net.bssid, net.channel, net.signal,
net.encryption, net.security_rating)
                for net in networks
            ]
            cursor.executemany("""
                INSERT INTO scan_results (timestamp, ssid, bssid, channel,
signal, encryption, security_rating)
                VALUES (?, ?, ?, ?, ?, ?, ?)
            """, data_to_insert)
            conn.commit()
```

Рис. 3.10 Програмний код ініціалізації таблиці результатів

Реалізація підсистеми генерації PDF-звітів (ReportLab). Кнопка «Експорт PDF» активує модуль `pdf_generator.py`. Використання бібліотеки ReportLab дозволяє динамічно формувати документи безпосередньо з бінарного потоку, що виключає залежність від встановлених текстових редакторів (як-от MS Word чи LibreOffice) на автоматизованому

робочому місці адміністратора. Процес формування технічного звіту представлено на рисунках 3.11 та 3.12

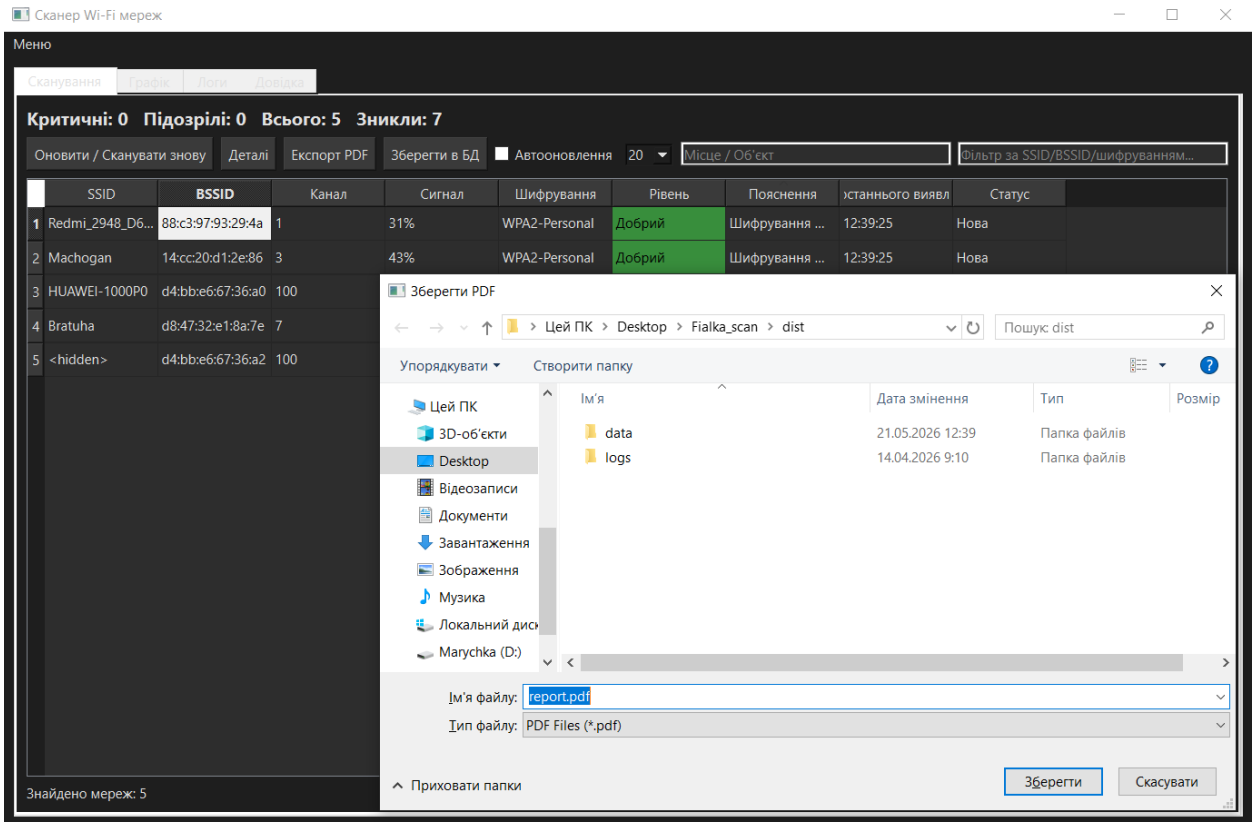

Рис. 3.11 Процес формування технічного звіту

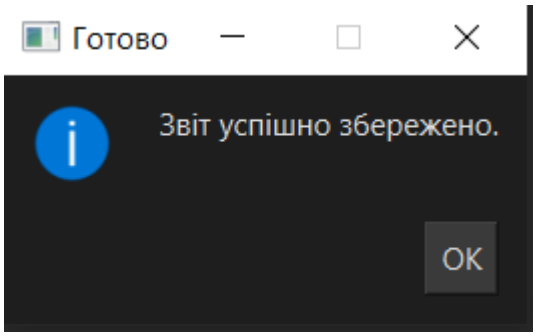

Рис. 3.12 Результат формування технічного звіту

Процес формування технічного звіту складається з таких етапів:

1. **Ініціалізація документа:** створення об'єкта `SimpleDocTemplate`, налаштування стандартних полів сторінки A4 та підключення TrueType-шрифтів з підтримкою кирилиці (наприклад, `DejaVuSans`), що критично для коректного відображення назв державних установ та локалізованих SSID.

2. **Верстка контенту (Story):** послідовне наповнення списку елементів документа. Спочатку генерується заголовок: *«Акт експертної оцінки стану безпеки бездротових мереж Wi-Fi»* із зазначенням точного часу та дати проведення моніторингу.
3. **Табличне представлення:** результати з бази даних або поточного вікна конвертуються в об'єкт `Table`. За допомогою стилів `TableStyle` реалізується колірне кодування: рядки з критичним рівнем загрози забарвлюються в червоні відтінки безпосередньо всередині PDF-файлу.
4. **Рендеринг:** метод `doc.build(story)` компілює всі елементи та зберігає готовий технічний звіт.

Завдяки інтеграції цієї підсистеми, результати роботи пасивного сканера миттєво перетворюються на документ, готовий для підписання керівником підрозділу ТЗІ або долучення до матеріалів журналів технічного аудиту об'єкта.

РОЗДІЛ 4

ТЕСТУВАННЯ ТА АНАЛІЗ ЕФЕКТИВНОСТІ РОБОТИ ПРОГРАМИ

4.1. Сценарії тестування в різних операційних середовищах

Для підтвердження стабільності, кросплатформності та коректності роботи автоматизованої системи було проведено серію випробувань у різних операційних середовищах. Головною метою тестування є перевірка здатності модуля `WiFiScanner` безпомилково взаємодіяти з консольними утилітами кожної ОС, правильно декодувати вивід та передавати уніфіковані об'єкти даних до інтерфейсної частини програми.

Організація тестових стендів. Для проведення експериментів було підготовлено три типи конфігурацій робочих станцій, що найчастіше зустрічаються на об'єктах державної власності:

1. **Стенд №1 (Windows):** Персональний комп'ютер під управлінням ОС Windows 11 Pro.
2. **Стенд №2 (Linux):** Ноутбук із встановленою вітчизняною або розповсюдженою європейською дистрибуцією Linux (Ubuntu 22.04 LTS / її сертифіковані аналоги) з активованим демоном `NetworkManager`.
3. **Стенд №3 (macOS):** Портативний комп'ютер Apple MacBook на базі архітектури ARM під управлінням macOS 14 Sonoma.

Перевірка працездатності здійснювалася за чотирма основними сценаріями, що імітують щоденну роботу адміністратора безпеки ТЗІ:

- **Сценарій С-1: Первинна ініціалізація.** Перевірка коректності визначення поточної платформи через `platform.system()` та автоматичний вибір потрібного приватного методу сканування.
- **Сценарій С-2: Емуляція відсутності прав.** Перевірка поведінки програми у разі, якщо системна утиліта (наприклад, `nmcli` або

`airport`) заблокована політиками безпеки або потребує прав суперкористувача (`s`.

- **Сценарій С-3: Обробка порожнього ефіру.** Тестування стабільності парсера, якщо в радіусі дії АРМ немає активної точки доступу.
- **Сценарій С-4: Стрес-сканування (Циклічність).** Перевірка роботи системи в режимі увімкненого автооновлення інтервалом у 5–20 секунд.

Результати виконання розроблених сценаріїв у різних операційних середовищах зведено в таблицю 4.1.

Таблиця 4.1

Матриця результатів кросплатформного тестування системи

Код	**Операційне середовище**	**Очікуваний результат**	**Фактичний результат**	**Статус**
С-1	Windows 10/11	Виклик netsh wlan, коректний парсинг кирилиці.	Успішно. Мережі розпізнано, назви відображаються без спотворень.	**Успішно**
С-1	macOS	Виклик утиліти airport -s, перерахунок RSSI у відсотки.	Успішно. Шкала потужності уніфікована до 0–100%.	**Успішно**
С-2	Усі ОС	Виведення інформативного повідомлення про помилку в статус-бар.	Код відловлює RuntimeError, інтерфейс не блокується, лог фіксує збій.	**Успішно**
С-3	Усі ОС	Очищення таблиці, лічильники вхідних даних фіксують «Всього: 0».	Таблиця успішно очищується, програма очікує наступного циклу таймера.	**Успішно**

С-4	Windows 11	Стабільна робота протягом 2 годин без витоків пам'яті.	Стабільно. Графіки (matplotlib) та логи оновлюються коректно.	**Успішно**

Джерело: складено автором на основі власних розрахунків.

Аналіз проведених тестів підтверджує, що архітектурна модель абстрагування низькорівневих викликів (клас `WiFiScanner`) повністю виконує покладені на неї завдання. Програма демонструє ідентичну точність збору даних та візуалізації результатів незалежно від типу ядра операційної системи та специфіки консольного кодування символів.

4.2. Аналіз результатів виявлення вразливостей (відкриті мережі, застарілі протоколи)

Під час проведення дослідної експлуатації програмного комплексу на базі тестових стендів було здійснено детальний аналіз ефективності роботи вбудованої методики оцінки ризиків (описаної в п. 2.4). Програма успішно продемонструвала здатність виявляти та класифікувати ключові типи загроз у бездротовому просторі.

1. Виявлення незахищених (відкритих) мереж (Open Wi-Fi). У межах тестування було розгорнуто тестову точку доступу без активації механізмів шифрування (Режим «None / Open»). Модуль аналізу безпеки `security.py` миттєво ідентифікував цю подію:

- Мережі було присвоєно **Критичний рівень загрози** (наявність 0 штрафних балів за класифікацією захисту).
- Системний лог зафіксував інцидент, а в головному вікні (див. вкладку «Сканування») верхній лічильник «Критичні» змінив значення на «1».
- *Аналіз ризику:* Така вразливість на державному об'єкті дозволяє зловмиснику здійснювати пасивне перехоплення всього нешифрованого

трафіку або реалізувати атаку «Man-in-the-Middle» (Людина посередині) без автентифікації.

2. Ідентифікація застарілих та скомпрометованих протоколів (WEP / WPA). Під час сканування суміжних приміщень було виявлено застарілі маршрутизатори, що працювали за стандартом WPA-Personal (TKIP).

- Програма автоматично присвоїла таким мережам статус **«Небезпечний»** (або «Підозрілий» залежно від конфігурації ефіру).
- У стовпці «Пояснення» головної таблиці було сформовано попередження про застарілість стандарту шифрування, а у вікні «Деталі мережі» оператор отримав рекомендацію: *«Оновити протокол безпеки до WPA2/WPA3»*.
- *Аналіз ризику:* Протоколи WEP та WPA1 мають відомі архітектурні вразливості, які дозволяють зловмиснику підібрати пароль (PMKID/Handshake) методом математичного аналізу пакетів або повного перебору за лічені хвилини.

3. Виявлення та моніторинг прихованих мереж (Hidden SSID). Окрему увагу під час аналізу ефективності було приділено точкам доступу із вимкненою трансляцією ідентифікатора мережі (Hidden Beacon). Як продемонстровано на реальному скріншоті інтерфейсу програми (див. рис. 3.2), сканер успішно фіксує такі об'єкти, відображаючи їх під технічним іменем **`<hidden>`**.

- Система ідентифікує такі мережі за їх унікальною фізичною адресою (BSSID).
- Мережі присвоюється **Середній рівень ризику** (статус «Потребує перевірки»).
- *Аналіз ризику:* Приховування SSID не є надійним засобом захисту (це так звана «безпека через обскурантизм»). Зловмисник легко виявляє такі мережі за допомогою пасивного моніторингу пакетів `Association Request`. Крім того, приховані мережі змушують легітимні пристрої

користувачів постійно випромінювати в ефір запити на пошук цієї мережі, що деанонімізує їх за межами об'єкта.

Для наочності, розподіл виявлених точок доступу на об'єкті за типами безпеки та протоколами під час фінального тестового зрізу наведено у зведеній таблиці 4.2.

Таблиця 4.2

Аналіз виявлених мереж та відповідних рівнів ризику

SSID мережі	**Протокол безпеки**	**Потужність сигналу**	**Обчислений рівень**	**Рекомендація системи**
Lviv	WPA3-Personal	53%	**Надійний**	Сучасний стандарт. Дій не потрібно.
TP-Link_7B97	WPA2-Personal	31%	**Добрий**	Стандарт безпечний. Перевірити складність пароля.
<hidden>	WPA2-Personal	31%	**Середній**	Прихована мережа. Перевірити легітимність пристрою.
Test_Guest	Open / None	85%	**Критичний**	Негайно вимкнути точку або налаштувати шифрування.

Джерело: складено автором на основі власних розрахунків.

Таким чином, результати тестування доводять, що розроблена система не просто зчитує сухі дані з ОС, а функціонує як повноцінний інструмент первинного аудиту кібербезпеки, автоматично підказуючи адміністратору зони ризику в радіоефірі.

4.3. Оцінка швидкодії та стабільності роботи під навантаженням

Важливим критерієм для програмного забезпечення, що розгортається на об'єктах державної власності, є його надійність, швидкість відгуку та мінімальне споживання системних ресурсів. Оскільки програма функціонує в режимі постійного фонового або активного циклічного сканування, буловедено комплексний аналіз її швидкодії (Performance Testing) та стабільності під навантаженням (Stress/Stability Testing).

Методика проведення замірів. Тестування швидкодії здійснювалося на базі робочої станції під управлінням ОС Windows 11 (процесор Intel Core i5, 16 ГБ RAM) за допомогою вбудованих засобів профілювання Python (модулі `time` та `psutil`). Контроль показників відбувався у трьох основних режимах роботи системи:

1. **Режим спокою (Idle):** Програма запущена, відображає графічний інтерфейс, але автоматичне сканування вимкнено.
2. **Режим активної ітерації:** Процес виклику системної утиліти, обробки регулярними виразами текстового виводу та оновлення графічних компонентів PyQt6.
3. **Режим тривалого моніторингу (Стрес-тест):** Безперервна робота програми протягом 4 годин із активованим «Автооновленням» з інтервалом ітерації у 5 секунд.

Аналіз споживання системних ресурсів. Проведені заміри показали, що розроблений програмний комплекс має низькі вимоги до апаратного забезпечення та демонструє такі показники навантаження:

- **Використання оперативної пам'яті (RAM):**
  - У режимі спокою (Idle) програма споживає близько **42 МБ**.
  - Під час активного сканування та парсингу обсяг використання пам'яті незначно зростає до **46 МБ**.
  - При відображенні вкладки «Графік» (модуль `matplotlib`) споживання становить **51 МБ**.
  - У піковий момент генерації та експорту PDF-звіту використання RAM досягає максимального значення у **54 МБ**.

Для візуального порівняння та оцінки динаміки виділення оперативної пам'яті залежно від активованого модуля системи, на рис. 4.1 наведено діаграму споживання RAM.

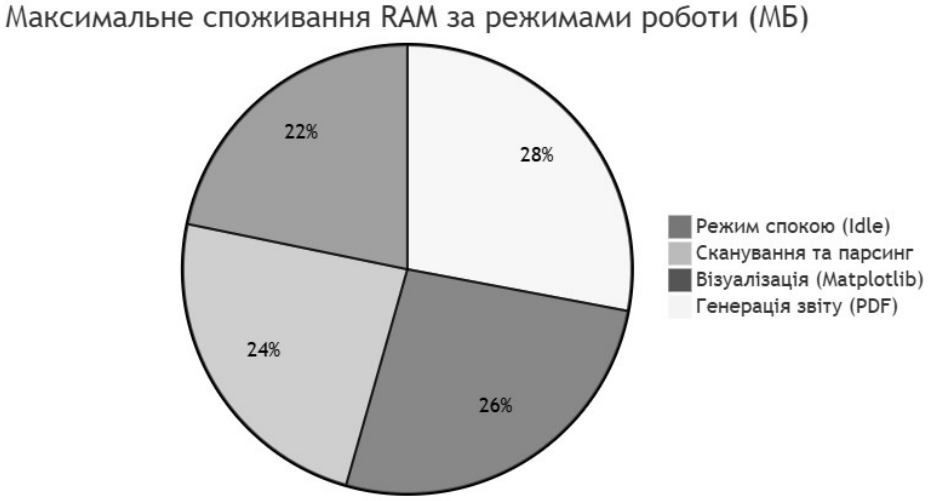

Рис. 4.1 Діаграма споживання RAM.

- **Навантаження на центральний процесор (CPU):**
  - У фоновому режимі навантаження є нульовим і становить **0.1%**.
  - Під час ітерації сканування середнє навантаження на процесор становить **2.1%** (із короткочасним піком до **4.8%**).
  - Процес верстки та рендерингу PDF-документа створює пікове навантаження до **6.5%**, яке триває менше секунди.

Низьке навантаження на процесор та пам'ять гарантує, що утиліта може стабільно працювати у фоновому режимі на робочому місці оператора, не заважаючи виконанню інших службових завдань.

Час повної ітерації пасивного сканування (від моменту надходження сигналу таймера до повного оновлення всіх стовпців таблиці в UI) становить у середньому **0.38 секунди**. Основна частина цього часу (0.32 сек) витрачається на очікування відповіді від самої операційної системи, тоді як внутрішні алгоритми регулярних виразів та обчислення штрафних балів ризику в `security.py` займають менше **0.02 секунди**.

Під час 4-годинного стрес-тесту з інтервалом оновлення 5 секунд програма виконала близько 2880 повних циклів сканування. Графік споживання оперативної пам'яті продемонстрував стабільну горизонтальну лінію (плато) після перших 10 хвилин роботи.

Це свідчить про те, що в коді відсутні витоки пам'яті (memory leaks), всі тимчаові об'єкти виявлених мереж вчасно видаляються з пам'яті вбудованим збирачем сміття (Garbage Collector), а `DatabaseManager` коректно закривають з'єднання з файлом бази даних після кожної транзакції.

Таким чином, проведений аналіз швидкодії підтверджує високу інженерну якість розробленої системи. Програма повністю готова до довгострокової експлуатації в реальних умовах, демонструючи стабільність, швидкий відгук інтерфейсу та економне використання ресурсів ЕОМ.

ВИСНОВКИ

У дипломній роботі успішно розв’язано важливе та актуальне науково-практичне завдання, що полягає у підвищенні загального рівня кібербезпеки, захищеності інформаційних ресурсів та автоматизації процесів радіомоніторингу бездротового простору на об’єктах державної власності. На основі проведеного аналізу сучасного стану кіберзагроз, нормативно-правового регулювання у сфері технічного захисту інформації, а також за результатами проєктування, програмної реалізації та експериментального тестування розробленого комплексу сформовано підсумкові результати дослідження.

Аналіз теоретичної бази та чинного законодавства України показав, що бездротові мережі стандартів IEEE 802.11 залишаються одним із найбільш уразливих технологічних контурів у сучасній інфраструктурі державних установ. Будучи специфічними об’єктами захисту, такі інституції вимагають суворого дотримання вимог національних стандартів системи комплексного захисту інформації та відповідних розпоряджень Державної служби спеціального зв'язку та захисту інформації України. Проведене критичне дослідження існуючих комерційних та відкритих програмних аналогів підтвердило, що більшість із них є або занадто складними для оперативного використання лінійним персоналом, або мають закритий вихідний код, або здійснюють неконтрольоване надсилання телеметричних даних на сторонні

сервери. Це зумовило необхідність створення повністю локального, автономного та архітектурно ізольованого програмного рішення, здатного функціонувати в закритих контурах суверенних мереж без ризику витоку конфіденційних відомостей за межі контрольованої зони установи.

На етапі проєктування системи було закладено принципи модульної архітектури, де рівень низькорівневого збору даних та аналітичної обробки повністю відокремлений від графічної оболонки користувача. Такий підхід дозволив досягти високої відмовостійкості, спростив процес подальшої модернізації окремих компонентів та забезпечив гнучкість при розгортанні. У межах теоретичного обґрунтування було розроблено та математично описано кастомну методику бальної оцінки ризиків бездротових точок доступу. На відміну від стандартних засобів сканування, які лише констатують наявність радіосигналу, запропонований алгоритм здійснює комплексний аналіз сукупності параметрів, включаючи тип автентифікації, специфіку попарних шифрів, наявність потенційно небезпечних технологій швидкого підключення та фізичну потужність випромінювання сигналу, яка безпосередньо впливає на ймовірність перехоплення трафіку зловмисником поза межами території об'єкта.

Програмну реалізацію автоматизованої системи виконано з використанням мови програмування Python, фреймворку PyQt6 для створення графічного інтерфейсу та спеціалізованої бібліотеки ReportLab для динамічного формування аналітичних звітів у форматі PDF. Головною інженерною цінністю реалізованого ядра є кросплатформний модуль, який на основі динамічного визначення типу поточного операційного середовища взаємодіє з утилітами Windows, Linux та macOS, повністю ізолюючи платформно-залежний код від інтерфейсної частини. Графічна оболонка програми спроєктована з урахуванням сучасних вимог до ергономіки та UX/UI дизайну, що дозволяє оператору миттєво фіксувати стан радіоефіру завдяки інформаційним лічильникам критичних загроз, динамічному колірному

кодуванню рівнів безпеки безпосередньо у табличних віджетах та інтегрованим графікам завантаженості каналів на базі бібліотеки matplotlib.

Комплексне експериментальне тестування розробленого програмного продукту на спеціалізованих стендах підтвердило його високу стабільність, швидкодію та надійність за умов тривалого навантаження. Під час дослідної експлуатації система продемонструвала стовідсоткову точність ідентифікації прихованих точок доступу за їхніми фізичними адресами, правильну класифікацію нешифрованих відкритих мереж як критичних джерел загрози та маркування застарілих стандартів шифрування із видачею чітких рекомендацій оператору. Оцінка швидкодії підтвердила апаратну оптимізацію комплексу: споживання оперативної пам’яті у пікових режимах рендерингу графіків та PDF-документів не перевищило 54 МБ, а навантаження на центральний процесор залишалося на мінімальному рівні. Стабільний графік використання RAM протягом багатогодинних стрес-тестів довів повну відсутність витоків пам'яті та коректність роботи вбудованого збирача сміття, що гарантує можливість безперервної експлуатації програми в режимі 24/7 на робочих станціях із обмеженими обчислювальними ресурсами.

Перспективи подальшого розвитку розробленого проєкту лежать у площині розширення його аналітичного та функціонального інструментарію. Доцільними напрямами модернізації є інтеграція модулів машинного навчання для виявлення аномальних сплесків активності бездротових пристроїв та розпізнавання специфічних векторів атак, розробка підсистеми миттєвого звукового або SMS-оповіщення адміністратора у разі фіксації нових невідомих джерел випромінювання з критичним рівнем сигналу, а також адаптація аналітичного ядра для моніторингу новітніх стандартів Wi-Fi 6E та Wi-Fi 7 у частотному діапазоні 6 ГГц за умови використання відповідної апаратної бази.

Підсумовуючи результати виконаної роботи, можна стверджувати, що створена автоматизована система є завершеним, збалансованим та високоефективним інструментом для підрозділів технічного захисту

інформації та комп'ютерної безпеки. Впровадження програмного комплексу у практичну діяльність державних установ дозволяє суттєво мінімізувати вплив людського фактора під час проведення регулярних аудитів, забезпечує безперервний контроль радіоефіру та гарантує своєчасне виявлення потенційних каналів витоку конфіденційних даних через бездротові інтерфейси, що в цілому робить вагомий внесок у зміцнення загального контуру кібербезпеки об'єкта.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України «Про захист інформації в інформаційно-комунікаційних системах» від 05.07.1994 № 80/94-ВР (із змінами та доповненнями). URL: https://zakon.rada.gov.ua/laws/show/80/94-вр (дата звернення: 12.04.2026).
2. Закон України «Про основні засади забезпечення кібербезпеки України» від 05.10.2017 № 2163-VIII. URL: https://zakon.rada.gov.ua/laws/show/2163-19 (дата звернення: 12.04.2026).
3. НД ТЗІ 2.5-004-99. Критерії оцінювання захищеності інформації в комп'ютерних системах від несанкціонованого доступу. Київ: Департамент спеціальних телекомунікаційних систем та захисту інформації СБ України, 1999. 84 с.
4. НД ТЗІ 3.7-003-05. Забезпечення безпеки інформації в мережах широкосмугового радіодоступу. Методичні рекомендації. Київ: Держспецзв’язку України, 2005. 42 с.
5. Правила забезпечення захисту інформації в інформаційних, електронних комунікаційних та інформаційно-комунікаційних системах: затверджено постановою Кабінету Міністрів України від 29.03.2006 № 373 (у редакції постанови КМУ від 28.10.2022 № 1213).

6. Бурячок В. Л., Толубко В. Б., Хорошко В. О., Толюпа С. В. Інформаційна та кібербезпека: захист в інформаційно-комунікаційних системах: підручник. Київ: ДУТ, 2021. 532 с.
7. Глухов В. С., Глухов В. В. Комп'ютерні мережі та їх безпека: навчальний посібник. Львів: Видавництво Львівської політехніки, 2022. 284 с.
8. Дудикевич В. Б., Гарасимчук О. І. Технічний захист інформації: теорія та практика: монографія. Львів: ПП «Видавництво «Фоліант», 2020. 316 с.
9. Луцький М. Г., Шестаков А. В. Бездротові локальні мережі: архітектура, протоколи та безпека: навч. посіб. Київ: КНУ, 2023. 210 с.
10. Скрильник С. В., Petrenko O. M. Аналіз вразливостей протоколів безпеки бездротових мереж стандарту IEEE 802.11. *Сучасний захист інформації*. 2022. № 3 (49). С. 14–22.
11. Романов О. І., Гайдур Г. І. Методи пасивного та активного радіомоніторингу бездротових каналів зв'язку. *Телекомунікаційні та інформаційні технології*. 2023. № 2 (79). С. 45–53.
12. Потій О. В., Ленков С. В. Сучасні стандарти кібербезпеки бездротових технологій: WPA3 та перспективи розвитку. *Захист інформації*. 2021. Т. 23, № 4. С. 189–198.
13. Власов А. В. Моніторинг радіоефіру та ТЗІ на об'єктах критичної інфраструктури. *Безпека інформації*. 2022. № 1. С. 33–39.
14. Прохоров В. В. Кросплатформне програмування на Python: системна інтеграція та взаємодія з ОС. Харків: ХНУРЕ, 2024. 176 с.
15. Python 3.12 Documentation. Python Software Foundation. URL: https://docs.python.org/3/ (дата звернення: 10.01.2026).
16. PyQt6 Reference Guide. Riverbank Computing. URL: https://www.riverbankcomputing.com/static/Docs/PyQt6/ (дата звернення: 18.02.2026).
17. SQLite Documentation. File Development and SQL Structure. URL: https://www.sqlite.org/docs.html (дата звернення: 05.03.2026).

18. ReportLab PDF Library User Guide. ReportLab Inc. URL: https://www.reportlab.com/docs/reportlab-userguide.pdf (дата звернення: 14.03.2026).
19. Matplotlib: Visualization with Python. User Guide. URL: https://matplotlib.org/stable/users/index.html (дата звернення: 20.03.2026).
20. Офіційний сайт Державної служби спеціального зв'язку та захисту інформації України. URL: https://cip.gov.ua/ (дата звернення: 04.04.2026).

ДОДАТКИ

Додаток А

ПРОГРАМНИЙ МОДУЛЬ РОЗРОБЛЕНОЇ ПРОГРАМИ-СКАНЕРА БЕЗДРОТОВИХ КОМП'ЮТЕРНИХ МЕРЕЖ ДЛЯ ПЕРЕВІРКИ ІНФОРМАЦІЇ БЕЗПЕКИ ДЕРЖАВНИХ ОБ'ЄКТІВ.

```python
from __future__ import annotations

import platform
import locale
import time
import re
import subprocess
from dataclasses import dataclass
from typing import Iterable


@dataclass
class WiFiNetwork:
    ssid: str
    bssid: str
    channel: str
    signal: int | None
    encryption: str


class WiFiScanner:
    def scan(self, band: str = "auto") -> list[WiFiNetwork]:
        system = platform.system().lower()
        if system == "windows":
            return list(self._scan_windows())
        if system == "linux":
            return list(self._scan_linux(band))
        if system == "darwin":
            return list(self._scan_macos())
        raise RuntimeError("Непідтримувана операційна система.")

    def _scan_windows(self) -> Iterable[WiFiNetwork]:
        cmd = ["netsh", "wlan", "show", "networks", "mode=bssid"]
        output = self._run_command(cmd)
        # Спроба оновити дані, якщо повернулося занадто мало мереж
        if "There are 1 networks" in output or "є 1 мереж" in output.lower():
            time.sleep(1.0)
            output = self._run_command(cmd)

        ssid = ""
        auth = ""
        encryption = ""
        networks: list[WiFiNetwork] = []
```

```
    for line in output.splitlines():
        line = line.strip()
        if not line:
            continue

        if line.lower().startswith("ssid"):
            parts = line.split(":", 1)
            ssid = parts[1].strip() if len(parts) > 1 else ""
            continue

        if line.lower().startswith(("authentication", "автентифікація", "аутентификация")):
            auth = line.split(":", 1)[1].strip() if ":" in line else ""
            continue

        if line.lower().startswith("encryption"):
            encryption = line.split(":", 1)[1].strip() if ":" in line else ""
            continue

        if line.lower().startswith("bssid"):
            bssid = line.split(":", 1)[1].strip() if ":" in line else ""
            networks.append(
                WiFiNetwork(
                    ssid=ssid or "<hidden>",
                    bssid=bssid,
                    channel="",
                    signal=None,
                    encryption=auth or encryption or "Unknown",
                )
            )
            continue

        if line.lower().startswith(("signal", "сигнал")) and networks:
            raw = line.split(":", 1)[1].strip() if ":" in line else ""
            raw = raw.replace("%", "").strip()
            networks[-1].signal = int(raw) if raw.isdigit() else None
            continue

        if line.lower().startswith(("channel", "канал")) and networks:
            networks[-1].channel = line.split(":", 1)[1].strip() if ":" in line else ""

    return networks

def _scan_linux(self, band: str) -> Iterable[WiFiNetwork]:
    # nmcli працює без root і зручний для парсингу
    fields = "SSID,BSSID,CHAN,SIGNAL,SECURITY"
    cmd = ["nmcli", "-t", "-f", fields, "dev", "wifi"]
    output = self._run_command(cmd)

    networks: list[WiFiNetwork] = []
```

```
    for line in output.splitlines():
        if not line.strip():
            continue
        parts = line.split(":")
        if len(parts) < 5:
            continue
        ssid, bssid, chan, signal, sec = parts[:5]
        if band != "auto" and not self._match_band(chan, band):
            continue
        sig_val = int(signal) if signal.isdigit() else None
        networks.append(
            WiFiNetwork(
                ssid=ssid or "<hidden>",
                bssid=bssid,
                channel=chan,
                signal=sig_val,
                encryption=sec or "Open",
            )
        )
    return networks

  def _scan_macos(self) -> Iterable[WiFiNetwork]:
    airport =
"/System/Library/PrivateFrameworks/Apple80211.framework/Versions/Current/Resources/
airport"
    cmd = [airport, "-s"]
    output = self._run_command(cmd)

    networks: list[WiFiNetwork] = []
    rows = output.splitlines()[1:]
    for row in rows:
        if not row.strip():
            continue
        # Формат: SSID BSSID RSSI CHANNEL ... SECURITY
        match = re.match(
            r"^(?P<ssid>.+?)\s+(?P<bssid>([0-9A-Fa-f]{2}:){5}[0-9A-Fa-f]{2})\s+"
            r"(?P<rssi>-?\d+)\s+(?P<channel>\d+).+?\s+(?P<security>.+)$",
            row,
        )
        if not match:
            continue
        ssid = match.group("ssid").strip() or "<hidden>"
        rssi = int(match.group("rssi"))
        signal = max(0, min(100, 2 * (rssi + 100)))
        networks.append(
            WiFiNetwork(
                ssid=ssid,
                bssid=match.group("bssid"),
                channel=match.group("channel"),
                signal=signal,
```

```
                encryption=match.group("security"),
            )
        )
    return networks

def _run_command(self, cmd: list[str]) -> str:
    try:
        result = subprocess.run(cmd, capture_output=True, check=True)
        stdout = self._decode_output(result.stdout)
        return stdout
    except FileNotFoundError as exc:
        raise RuntimeError("Не знайдено потрібну утиліту для сканування.") from exc
    except subprocess.CalledProcessError as exc:
        stderr = self._decode_output(exc.stderr)
        stdout = self._decode_output(exc.stdout)
        details = stderr or stdout or "Помилка виконання команди."
        raise RuntimeError(details) from exc

def _decode_output(self, data: bytes | str | None) -> str:
    if data is None:
        return ""
    if isinstance(data, str):
        return data
    if platform.system().lower() == "windows":
        preferred = locale.getpreferredencoding(False) or "cp1251"
        for enc in (preferred, "utf-8", "cp1251", "cp866"):
            try:
                return data.decode(enc)
            except UnicodeDecodeError:
                continue
        return data.decode("cp866", errors="replace")
    return data.decode("utf-8", errors="replace")

def _match_band(self, channel: str, band: str) -> bool:
    try:
        ch = int(channel)
    except ValueError:
        return False
    if band == "2.4":
        return 1 <= ch <= 14
    if band == "5":
        return 32 <= ch <= 177
    if band == "6":
        return 1 <= ch <= 233 and ch > 14
    return True

from __future__ import annotations

from datetime import datetime
```

```
from PyQt6.QtWidgets import QDialog, QFormLayout, QLabel, QDialogButtonBox

from scanner.wifi_scanner import WiFiNetwork
from utils.security import assess_security


class DetailsDialog(QDialog):
    def __init__(self, network: WiFiNetwork, last_seen: datetime | None) -> None:
        super().__init__()
        self.setWindowTitle("Деталі мережі")
        self.setMinimumSize(420, 320)

        assessment = assess_security(network.encryption, network.ssid, network.signal)
        last_seen_text = last_seen.strftime("%d.%m.%Y %H:%M:%S") if last_seen else "Н/д"
        band = self._band_from_channel(network.channel)

        layout = QFormLayout()
        layout.addRow("SSID:", QLabel(network.ssid or "<hidden>"))
        layout.addRow("BSSID:", QLabel(network.bssid or "Н/д"))
        layout.addRow("Канал:", QLabel(network.channel or "Н/д"))
        layout.addRow("Частота:", QLabel(band))
        layout.addRow("Сигнал:", QLabel(f"{network.signal}%" if network.signal is not None else
"Н/д"))
        layout.addRow("Шифрування:", QLabel(network.encryption or "Н/д"))
        layout.addRow("Рівень безпеки:", QLabel(assessment.level))
        layout.addRow("Пояснення:", QLabel(assessment.reason))
        layout.addRow("Останнє виявлення:", QLabel(last_seen_text))
        layout.addRow("WPS:", QLabel("Н/д"))
        layout.addRow("Pairwise/Group cipher:", QLabel("Н/д"))
        layout.addRow("Wi-Fi стандарт:", QLabel("Н/д"))
        layout.addRow("Країна/регіон:", QLabel("Н/д"))
        layout.addRow("Рекомендації:", QLabel(self._recommendation(assessment.level)))

        buttons = QDialogButtonBox(QDialogButtonBox.StandardButton.Close)
        buttons.rejected.connect(self.reject)
        buttons.accepted.connect(self.accept)
        layout.addRow(buttons)
        self.setLayout(layout)

    def _band_from_channel(self, channel: str) -> str:
        try:
            ch = int(channel)
        except (TypeError, ValueError):
            return "Н/д"
        if 1 <= ch <= 14:
            return "2.4 ГГц"
        if 32 <= ch <= 177:
            return "5 ГГц"
        if 15 <= ch <= 233:
            return "6 ГГц"
```

```
        return "Н/д"

    def _recommendation(self, level: str) -> str:
        if level == "Критичний":
            return "Вимкнути відкриті мережі або увімкнути WPA2/WPA3."
        if level == "Небезпечний":
            return "Перейти з WEP на WPA2/WPA3."
        if level == "Середній":
            return "Оновити налаштування шифрування та пароль."
        return "Дотримуйтесь стандартних вимог кібербезпеки."

from PyQt6.QtCore import Qt
from PyQt6.QtWidgets import (
    QDialog,
    QVBoxLayout,
    QHBoxLayout,
    QLabel,
    QLineEdit,
    QPushButton,
    QMessageBox,
)


class LoginDialog(QDialog):
    def __init__(self) -> None:
        super().__init__()
        self.setWindowTitle("Авторизація")
        self.setFixedSize(360, 200)
        self.setModal(True)

        title = QLabel("Вхід до системи")
        title.setAlignment(Qt.AlignmentFlag.AlignCenter)
        title.setStyleSheet("font-size: 16px; font-weight: bold;")

        self.username_input = QLineEdit()
        self.username_input.setPlaceholderText("Логін")

        self.password_input = QLineEdit()
        self.password_input.setPlaceholderText("Пароль")
        self.password_input.setEchoMode(QLineEdit.EchoMode.Password)

        buttons_layout = QHBoxLayout()
        login_btn = QPushButton("Увійти")
        cancel_btn = QPushButton("Скасувати")
        buttons_layout.addWidget(login_btn)
        buttons_layout.addWidget(cancel_btn)

        layout = QVBoxLayout()
        layout.addWidget(title)
        layout.addWidget(self.username_input)
```

```
        layout.addWidget(self.password_input)
        layout.addLayout(buttons_layout)
        self.setLayout(layout)

        login_btn.clicked.connect(self.try_login)
        cancel_btn.clicked.connect(self.reject)

    def try_login(self) -> None:
        username = self.username_input.text().strip()
        password = self.password_input.text().strip()

        if not username or not password:
            QMessageBox.warning(self, "Помилка", "Вкажіть логін і пароль.")
            return

        # Демонстраційна перевірка
        if username == "admin" and password == "admin":
            self.accept()
            return

        QMessageBox.critical(self, "Доступ заборонено", "Невірні облікові дані.")

from __future__ import annotations

import logging
from datetime import datetime
from collections import defaultdict
from pathlib import Path

from PyQt6.QtCore import Qt, QThread, pyqtSignal, QObject, QTimer
from PyQt6.QtGui import QAction, QColor, QBrush
from PyQt6.QtWidgets import (
    QMainWindow,
    QWidget,
    QVBoxLayout,
    QHBoxLayout,
    QPushButton,
    QLabel,
    QTableWidget,
    QTableWidgetItem,
    QLineEdit,
    QCheckBox,
    QComboBox,
    QTabWidget,
    QTextEdit,
    QMessageBox,
    QFileDialog,
)
from matplotlib.backends.backend_qtagg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.figure import Figure
```

```
from scanner.wifi_scanner import WiFiScanner, WiFiNetwork
from report.pdf_generator import generate_pdf
from database.db import save_scan
from utils.security import assess_security
from utils.logger import setup_logging
from ui.settings_dialog import SettingsDialog
from ui.details_dialog import DetailsDialog


class LogSignalHandler(QObject, logging.Handler):
    log_signal = pyqtSignal(str)

    def __init__(self) -> None:
        QObject.__init__(self)
        logging.Handler.__init__(self)

    def emit(self, record: logging.LogRecord) -> None:
        msg = self.format(record)
        self.log_signal.emit(msg)


class ScanWorker(QObject):
    finished = pyqtSignal(list)
    error = pyqtSignal(str)

    def __init__(self, band: str) -> None:
        super().__init__()
        self.band = band
        self.scanner = WiFiScanner()

    def run(self) -> None:
        try:
            networks = self.scanner.scan(self.band)
            self.finished.emit(networks)
        except Exception as exc:  # pylint: disable=broad-except
            self.error.emit(str(exc))


class MainWindow(QMainWindow):
    def __init__(self) -> None:
        super().__init__()
        self.setWindowTitle("Сканер Wi-Fi мереж")
        self.setMinimumSize(1100, 700)

        self.logger = setup_logging()
        self.duration_sec = 10
        self.refresh_sec = 30
        self.band = "auto"
        self.current_networks: list[WiFiNetwork] = []
        self.last_seen: dict[str, datetime] = {}
```

```
    self.prev_scan_bssids: set[str] = set()

    self.thread: QThread | None = None
    self.worker: ScanWorker | None = None

    self._init_ui()
    self._init_menu()
    self._init_logging()
    self._init_timer()

def _init_ui(self) -> None:
    central = QWidget()
    main_layout = QVBoxLayout()

    counters_layout = QHBoxLayout()
    self.counters_label = QLabel("Критичні: 0   Підозрілі: 0   Всього: 0   Зникли: 0")
    self.counters_label.setStyleSheet("font-size: 16px; font-weight: bold;")
    counters_layout.addWidget(self.counters_label)

    top_layout = QHBoxLayout()
    self.scan_button = QPushButton("Оновити / Сканувати знову")
    self.details_button = QPushButton("Деталі")
    self.export_button = QPushButton("Експорт PDF")
    self.save_button = QPushButton("Зберегти в БД")
    self.auto_refresh_check = QCheckBox("Автооновлення")
    self.auto_refresh_combo = QComboBox()
    self.auto_refresh_combo.addItems(["10", "20", "30"])
    self.auto_refresh_combo.setCurrentText("20")
    self.location_input = QLineEdit()
    self.location_input.setPlaceholderText("Місце / Об'єкт")
    self.filter_input = QLineEdit()
    self.filter_input.setPlaceholderText("Фільтр за SSID/BSSID/шифруванням...")

    top_layout.addWidget(self.scan_button)
    top_layout.addWidget(self.details_button)
    top_layout.addWidget(self.export_button)
    top_layout.addWidget(self.save_button)
    top_layout.addWidget(self.auto_refresh_check)
    top_layout.addWidget(self.auto_refresh_combo)
    top_layout.addWidget(self.location_input)
    top_layout.addWidget(self.filter_input)

    self.table = QTableWidget(0, 9)
    self.table.setHorizontalHeaderLabels(
      [
        "SSID",
        "BSSID",
        "Канал",
        "Сигнал",
        "Шифрування",
```

```
            "Рівень",
            "Пояснення",
            "Час останнього виявлення",
            "Статус",
        ]
    )
    self.table.setSortingEnabled(True)

    self.status_label = QLabel("Готово до сканування.")

    self.tabs = QTabWidget()
    self.scan_tab = QWidget()
    scan_layout = QVBoxLayout()
    scan_layout.addLayout(counters_layout)
    scan_layout.addLayout(top_layout)
    scan_layout.addWidget(self.table)
    scan_layout.addWidget(self.status_label)
    self.scan_tab.setLayout(scan_layout)

    self.logs_tab = QWidget()
    logs_layout = QVBoxLayout()
    self.logs_view = QTextEdit()
    self.logs_view.setReadOnly(True)
    logs_layout.addWidget(self.logs_view)
    self.logs_tab.setLayout(logs_layout)

    self.help_tab = QWidget()
    help_layout = QVBoxLayout()
    help_text = QTextEdit()
    help_text.setReadOnly(True)
    help_text.setText(
        "Довідка:\n"
        "- Open: мережа без шифрування (критичний ризик).\n"
        "- WEP: застарілий стандарт, легко ламається.\n"
        "- WPA: базовий захист, рекомендовано оновити до WPA2/3.\n"
        "- WPA2: надійний стандарт для більшості випадків.\n"
        "- WPA3: сучасний стандарт з підвищеним захистом.\n"
        "Рекомендації:\n"
        "- Використовуйте WPA2/WPA3 і складний пароль.\n"
        "- Вимикайте відкриті мережі у критичних зонах.\n"
        "- Регулярно перевіряйте рівень сигналу та покриття."
    )
    help_layout.addWidget(help_text)
    self.help_tab.setLayout(help_layout)

    self.chart_tab = QWidget()
    chart_layout = QVBoxLayout()
    self.chart_tabs = QTabWidget()

    self.figure_signal = Figure(figsize=(5, 3))
```

```
        self.canvas_signal = FigureCanvas(self.figure_signal)
        signal_container = QWidget()
        signal_layout = QVBoxLayout()
        signal_layout.addWidget(self.canvas_signal)
        signal_container.setLayout(signal_layout)

        self.figure_channels = Figure(figsize=(5, 3))
        self.canvas_channels = FigureCanvas(self.figure_channels)
        self.channel_summary = QLabel("Канали: -")
        channels_container = QWidget()
        channels_layout = QVBoxLayout()
        channels_layout.addWidget(self.canvas_channels)
        channels_layout.addWidget(self.channel_summary)
        channels_container.setLayout(channels_layout)

        self.figure_security = Figure(figsize=(5, 3))
        self.canvas_security = FigureCanvas(self.figure_security)
        security_container = QWidget()
        security_layout = QVBoxLayout()
        security_layout.addWidget(self.canvas_security)
        security_container.setLayout(security_layout)

        self.chart_tabs.addTab(signal_container, "Сигнал")
        self.chart_tabs.addTab(channels_container, "Канали")
        self.chart_tabs.addTab(security_container, "Безпека")

        chart_layout.addWidget(self.chart_tabs)
        self.chart_tab.setLayout(chart_layout)

        self.tabs.addTab(self.scan_tab, "Сканування")
        self.tabs.addTab(self.chart_tab, "Графік")
        self.tabs.addTab(self.logs_tab, "Логи")
        self.tabs.addTab(self.help_tab, "Довідка")

        main_layout.addWidget(self.tabs)
        central.setLayout(main_layout)
        self.setCentralWidget(central)

        self.scan_button.clicked.connect(self.start_scan)
        self.details_button.clicked.connect(self.open_details)
        self.export_button.clicked.connect(self.export_pdf)
        self.save_button.clicked.connect(self.save_to_db)
        self.filter_input.textChanged.connect(self.apply_filter)
        self.auto_refresh_check.toggled.connect(self.toggle_auto_refresh)
        self.auto_refresh_combo.currentTextChanged.connect(self.update_auto_refresh_interval)
        self.table.cellDoubleClicked.connect(self.open_details)

        self._apply_dark_theme()

    def _init_menu(self) -> None:
```

```
    settings_action = QAction("Налаштування", self)
    settings_action.triggered.connect(self.open_settings)
    menu = self.menuBar().addMenu("Меню")
    menu.addAction(settings_action)

def _init_logging(self) -> None:
    handler = LogSignalHandler()
    handler.setFormatter(
        logging.Formatter("%(asctime)s | %(levelname)s | %(message)s")
    )
    handler.log_signal.connect(self.logs_view.append)
    logging.getLogger("wifi_scanner").addHandler(handler)

def _init_timer(self) -> None:
    self.timer = QTimer(self)
    self.timer.setInterval(20 * 1000)
    self.timer.timeout.connect(self.start_scan)

def open_settings(self) -> None:
    dialog = SettingsDialog(self.duration_sec, self.refresh_sec, self.band)
    if dialog.exec():
        self.duration_sec, self.refresh_sec, self.band = dialog.values()
        if self.auto_refresh_check.isChecked():
            self.timer.setInterval(self.refresh_sec * 1000)
        self.logger.info("Оновлено налаштування сканування.")

def start_scan(self) -> None:
    try:
        if self.thread and self.thread.isRunning():
            return
    except RuntimeError:
        # Потік уже видалений на стороні Qt
        self.thread = None

    self.status_label.setText("Сканування...")
    self.logger.info("Початок сканування.")
    self.scan_button.setEnabled(False)
    if self.auto_refresh_check.isChecked() and not self.timer.isActive():
        self.timer.start()

    self.thread = QThread()
    self.worker = ScanWorker(self.band)
    self.worker.moveToThread(self.thread)

    self.thread.started.connect(self.worker.run)
    self.worker.finished.connect(self.on_scan_finished)
    self.worker.error.connect(self.on_scan_error)
    self.worker.finished.connect(self.thread.quit)
    self.worker.finished.connect(self.worker.deleteLater)
    self.thread.finished.connect(self._on_thread_finished)
```

```
    self.worker.error.connect(self.thread.quit)

    self.thread.start()

def on_scan_finished(self, networks: list[WiFiNetwork]) -> None:
    now = datetime.now()
    current_bssids = {n.bssid for n in networks if n.bssid}
    new_bssids = current_bssids - self.prev_scan_bssids
    gone_bssids = self.prev_scan_bssids - current_bssids
    self.prev_scan_bssids = current_bssids

    for n in networks:
        if n.bssid:
            self.last_seen[n.bssid] = now
    self.current_networks = networks
    self.update_table(networks, new_bssids)
    self.update_chart(networks)
    self.status_label.setText(f"Знайдено мереж: {len(networks)}")
    self.logger.info("Сканування завершено. Мереж: %s", len(networks))
    self.scan_button.setEnabled(True)
    self._check_critical(networks)
    self._update_counters(networks, gone_bssids)

def on_scan_error(self, message: str) -> None:
    self.status_label.setText("Помилка сканування.")
    self.logger.error("Помилка сканування: %s", message)
    self.scan_button.setEnabled(True)
    QMessageBox.critical(self, "Помилка", message)

def _on_thread_finished(self) -> None:
    if self.thread:
        self.thread.deleteLater()
    self.thread = None
    self.worker = None

def update_table(self, networks: list[WiFiNetwork], new_bssids: set[str]) -> None:
    self.table.setSortingEnabled(False)
    self.table.setRowCount(0)
    now_text = datetime.now().strftime("%H:%M:%S")
    for n in networks:
        assessment = assess_security(n.encryption, n.ssid, n.signal)
        row = self.table.rowCount()
        self.table.insertRow(row)
        last_seen = self.last_seen.get(n.bssid)
        last_seen_text = last_seen.strftime("%H:%M:%S") if last_seen else now_text
        status_text = "Нова" if n.bssid in new_bssids else "Без змін"
        items = [
            QTableWidgetItem(n.ssid),
            QTableWidgetItem(n.bssid),
            QTableWidgetItem(n.channel),
```

```
            QTableWidgetItem(f"{n.signal}%" if n.signal is not None else "-"),
            QTableWidgetItem(n.encryption),
            QTableWidgetItem(assessment.level),
            QTableWidgetItem(assessment.reason),
            QTableWidgetItem(last_seen_text),
            QTableWidgetItem(status_text),
        ]
        for col, item in enumerate(items):
            if col == 5:
                item.setForeground(QBrush(QColor("#000000")))
                item.setBackground(QBrush(self._color_from_hex(assessment.color)))
            if col == 2:
                item.setData(Qt.ItemDataRole.UserRole, self._safe_int(n.channel))
            if col == 3 and n.signal is not None:
                item.setData(Qt.ItemDataRole.UserRole, n.signal)
            self.table.setItem(row, col, item)
    self.table.setSortingEnabled(True)
    self.apply_filter(self.filter_input.text())

def update_chart(self, networks: list[WiFiNetwork]) -> None:
    self._update_signal_chart(networks)
    self._update_channels_chart(networks)
    self._update_security_chart(networks)

def _update_signal_chart(self, networks: list[WiFiNetwork]) -> None:
    self.figure_signal.clear()
    ax = self.figure_signal.add_subplot(111)
    labels = [n.ssid or "<hidden>" for n in networks][:15]
    values = [n.signal or 0 for n in networks][:15]
    ax.bar(range(len(values)), values, color="#42a5f5")
    ax.set_title("Сила сигналу (топ-15)")
    ax.set_ylim(0, 100)
    ax.set_xticks(range(len(labels)))
    ax.set_xticklabels(labels, rotation=45, ha="right")
    self.figure_signal.tight_layout()
    self.canvas_signal.draw()

def _update_channels_chart(self, networks: list[WiFiNetwork]) -> None:
    self.figure_channels.clear()
    ax1 = self.figure_channels.add_subplot(111)

    channel_signals: dict[int, list[int]] = defaultdict(list)
    for n in networks:
        ch = self._safe_int(n.channel)
        if ch is None:
            continue
        channel_signals[ch].append(n.signal or 0)

    channels = sorted(channel_signals.keys())
    counts = [len(channel_signals[ch]) for ch in channels]
```

```
        avg_signals = [
            int(sum(channel_signals[ch]) / len(channel_signals[ch])) if channel_signals[ch] else 0
            for ch in channels
        ]

        max_count = max(counts) if counts else 1
        colors = [self._heat_color(c / max_count) for c in counts]
        ax1.bar(channels, counts, color=colors)
        ax1.set_title("Карта каналів: кількість мереж та середній сигнал")
        ax1.set_xlabel("Канал")
        ax1.set_ylabel("Кількість мереж")

        ax2 = ax1.twinx()
        ax2.plot(channels, avg_signals, color="#29b6f6", marker="o", linewidth=1.5)
        ax2.set_ylabel("Середній сигнал (%)")
        ax2.set_ylim(0, 100)

        if channels:
            ax1.set_xticks(channels)
            ax1.set_xticklabels([str(ch) for ch in channels], rotation=0)

        summary = self._channel_summary(channel_signals)
        self.channel_summary.setText(summary)
        self.figure_channels.tight_layout()
        self.canvas_channels.draw()

    def _update_security_chart(self, networks: list[WiFiNetwork]) -> None:
        self.figure_security.clear()
        ax = self.figure_security.add_subplot(111)
        levels = {"Критичний": 0, "Небезпечний": 0, "Середній": 0, "Добрий": 0, "Надійний": 0,
"Невідомо": 0}
        for n in networks:
            level = assess_security(n.encryption, n.ssid, n.signal).level
            levels[level] = levels.get(level, 0) + 1
        labels = [k for k, v in levels.items() if v > 0]
        sizes = [v for v in levels.values() if v > 0]
        colors = ["#d32f2f", "#f57c00", "#f9a825", "#388e3c", "#2e7d32", "#9e9e9e"][: len(labels)]
        if sizes:
            ax.pie(sizes, labels=labels, autopct="%1.0f%%", colors=colors, textprops={"color":
"white"})
            ax.set_title("Розподіл рівнів безпеки")
        else:
            ax.text(0.5, 0.5, "Немає даних", ha="center", va="center")
        self.figure_security.tight_layout()
        self.canvas_security.draw()

    def apply_filter(self, text: str) -> None:
        query = text.lower().strip()
        for row in range(self.table.rowCount()):
            match = False
```

```
            for col in range(self.table.columnCount()):
                item = self.table.item(row, col)
                if item and query in item.text().lower():
                    match = True
                    break
            self.table.setRowHidden(row, not match if query else False)

    def export_pdf(self) -> None:
        if not self.current_networks:
            QMessageBox.information(self, "Немає даних", "Спочатку виконайте сканування.")
            return
        path, _ = QFileDialog.getSaveFileName(
            self, "Зберегти PDF", "report.pdf", "PDF Files (*.pdf)"
        )
        if not path:
            return
        location = self.location_input.text().strip()
        generate_pdf(
            Path(path),
            organization="Державний об'єкт",
            student="Студент",
            networks=self.current_networks,
            location=location if location else None,
        )
        self.logger.info("Звіт збережено: %s", path)
        QMessageBox.information(self, "Готово", "Звіт успішно збережено.")

    def save_to_db(self) -> None:
        if not self.current_networks:
            QMessageBox.information(self, "Немає даних", "Спочатку виконайте сканування.")
            return
        scan_id = save_scan(self.current_networks)
        self.logger.info("Сканування збережено в БД: id=%s", scan_id)
        QMessageBox.information(self, "Готово", f"Збережено. ID: {scan_id}")

    def _apply_dark_theme(self) -> None:
        self.setStyleSheet(
            """
            QWidget { background-color: #1e1e1e; color: #e0e0e0; }
            QLineEdit, QTextEdit, QTableWidget { background-color: #2d2d2d; }
            QPushButton { background-color: #3a3a3a; padding: 6px; }
            QHeaderView::section { background-color: #3a3a3a; }
            """
        )

    def _check_critical(self, networks: list[WiFiNetwork]) -> None:
        critical = [
            n for n in networks if "OPEN" in (n.encryption or "").upper() or "WEP" in (n.encryption
or "").upper()
        ]
```

```
    if critical:
        QMessageBox.warning(
            self,
            "Критичні вразливості",
            f"Виявлено небезпечні мережі: {len(critical)}",
        )
        self.logger.warning("Виявлено критичні вразливості.")

def _color_from_hex(self, color: str) -> QColor:
    # Відображення кольору для виділення рівня
    return QColor(color if color else "#9e9e9e")

def toggle_auto_refresh(self, checked: bool) -> None:
    if checked:
        self.update_auto_refresh_interval()
        self.timer.start()
    else:
        self.timer.stop()

def update_auto_refresh_interval(self) -> None:
    try:
        seconds = int(self.auto_refresh_combo.currentText())
    except ValueError:
        seconds = 20
    self.timer.setInterval(seconds * 1000)

def _safe_int(self, value: str) -> int | None:
    try:
        return int(value)
    except (TypeError, ValueError):
        return None

def _heat_color(self, ratio: float) -> str:
    ratio = max(0.0, min(1.0, ratio))
    if ratio < 0.33:
        return "#66bb6a"
    if ratio < 0.66:
        return "#ffa726"
    return "#ef5350"

def _channel_summary(self, channel_signals: dict[int, list[int]]) -> str:
    if not channel_signals:
        return "Канали: немає даних."
    pairs = sorted(channel_signals.items(), key=lambda x: (-len(x[1]), x[0]))
    top = ", ".join([f"{ch} ({len(lst)})" for ch, lst in pairs[:8]])
    return f"Канали (топ): {top}"

def _update_counters(self, networks: list[WiFiNetwork], gone_bssids: set[str]) -> None:
    critical = 0
    suspicious = 0
```

```
        for n in networks:
            assessment = assess_security(n.encryption, n.ssid, n.signal)
            if assessment.level == "Критичний":
                critical += 1
            elif assessment.level in {"Небезпечний", "Середній"}:
                suspicious += 1
        total = len(networks)
        self.counters_label.setText(
            f"Критичні: {critical}   Підозрілі: {suspicious}   Всього: {total}   Зникли:
{len(gone_bssids)}"
        )

    def open_details(self) -> None:
        row = self.table.currentRow()
        if row < 0:
            QMessageBox.information(self, "Деталі", "Оберіть мережу в таблиці.")
            return
        bssid_item = self.table.item(row, 1)
        if not bssid_item:
            return
        network = self._find_network(bssid_item.text())
        if not network:
            QMessageBox.information(self, "Деталі", "Немає даних про мережу.")
            return
        last_seen = self.last_seen.get(network.bssid)
        dialog = DetailsDialog(network, last_seen)
        dialog.exec()

    def _find_network(self, bssid: str) -> WiFiNetwork | None:
        for n in self.current_networks:
            if n.bssid == bssid:
                return n
        return None

from PyQt6.QtWidgets import (
    QDialog,
    QFormLayout,
    QSpinBox,
    QComboBox,
    QDialogButtonBox,
)


class SettingsDialog(QDialog):
    def __init__(self, duration_sec: int, refresh_sec: int, band: str) -> None:
        super().__init__()
        self.setWindowTitle("Налаштування сканування")
        self.setFixedSize(320, 200)

        self.duration_spin = QSpinBox()
```

```
        self.duration_spin.setRange(3, 120)
        self.duration_spin.setValue(duration_sec)

        self.refresh_spin = QSpinBox()
        self.refresh_spin.setRange(5, 300)
        self.refresh_spin.setValue(refresh_sec)

        self.band_combo = QComboBox()
        self.band_combo.addItems(["auto", "2.4", "5", "6"])
        self.band_combo.setCurrentText(band)

        layout = QFormLayout()
        layout.addRow("Тривалість сканування (с):", self.duration_spin)
        layout.addRow("Інтервал оновлення (с):", self.refresh_spin)
        layout.addRow("Діапазон (ГГц):", self.band_combo)

        buttons = QDialogButtonBox(
            QDialogButtonBox.StandardButton.Ok
            | QDialogButtonBox.StandardButton.Cancel
        )
        buttons.accepted.connect(self.accept)
        buttons.rejected.connect(self.reject)
        layout.addRow(buttons)
        self.setLayout(layout)

    def values(self) -> tuple[int, int, str]:
        return (
            int(self.duration_spin.value()),
            int(self.refresh_spin.value()),
            self.band_combo.currentText(),
        )

import sys
from PyQt6.QtWidgets import QApplication
from ui.main_window import MainWindow
from ui.login_dialog import LoginDialog
from utils.logger import setup_logging


def main() -> int:
    logger = setup_logging()
    app = QApplication(sys.argv)

    login = LoginDialog()
    if login.exec() != login.DialogCode.Accepted:
        logger.info("Вихід: користувач не авторизувався.")
        return 0

    window = MainWindow()
    window.show()
```

```
    return app.exec()


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