Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп’ютерних наук і технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти – бакалавр
за освітньо-професійною програмою
«Комп’ютерні науки»
зі спеціальності
122 – Комп’ютерні науки

тема роботи:

«Розробка сервісу комплексної перевірки порушення інформаційної безпеки»

Виконав студент гр. КН-21 _____ Коваленко І.С.

Керівник _____ Кумченко Ю.О.

Нормоконтроль _____ Маринич І. А.

Завідувач кафедри _____ Рубан С. А.

Кривий Ріг – 2025

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет: *інформаційних технологій*

Кафедра: *автоматизації, комп'ютерних наук і технологій*

Ступінь вищої освіти: *Бакалавр*

Спеціальність: *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ

Зав. кафедрою: *к.т.н. Рубан С.А.*

_____

« *29* » *січня* 2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

студентові групи КН-21 *Коваленко Ігорю Сергійовичу*

1. Тема кваліфікаційної роботи: *«Розробка сервісу комплексної перевірки порушення інформаційної безпеки»*

затверджено наказом по університету № *254с* від *13.05.2025 р.*

2. Термін здачі кваліфікаційної роботи: *06.06.2025 р.*

3. Склад кваліфікаційної роботи: *Пояснювальна записка обсягом 74с., додатки, презентація у Microsoft PowerPoint (13 слайдів) в електронному та друкованому вигляді*

4. Консультанти кваліфікаційної роботи:

Розділ 1-2 доц. Кумченко Ю.О.

Нормоконтроль доц. Маринич І. А.

5. Календарний план:

№	Етапи роботи	Термін виконання
1	*Вступ*	*18.03.25*
2	*Розділ 1*	*29.03.25*
3	*Розділ 2*	*20.05.25*
4	*Висновки*	*23.05.25*
5	*Оформлення кваліфікаційної роботи*	*23.05.25*
6	*Підготовка презентації та графічного матеріалу*	*01.06.25*
7	*Підготовка доповіді до захисту*	*02.06.25*

6. Дата видачі завдання: *28.01.2025р.*

Керівник _____ / Кумченко Ю.О./

7. Запевнення: *Я, Коваленко Ігор Сергійович, запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.*

Із чинним Положенням про академічну доброчесність Криворізького національного університету ознайомлений.

Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі умисних порушень робота не допускається до захисту або оцінюється незадовільно.

Студент _____ / Коваленко І.С./

АНОТАЦІЯ

Коваленко І.С. Розробка сервісу комплексної перевірки порушення інформаційної безпеки : кваліфікаційна робота бакалавра : 122 – Комп'ютерні науки. Кривий Ріг. Криворізький національний університет, 2025. 65 с.

Метою роботи є розробка веб-сервісу для комплексної перевірки порушень інформаційної безпеки, який забезпечує виявлення основних типів кіберзагроз і сприяє підвищенню рівня захисту персональних та корпоративних даних.

У вступі обґрунтовується актуальність теми роботи, сформульована мета та завдання для її досягнення.

У першому розділі проаналізовано сучасний стан інформаційної безпеки, основні типи кіберзагроз, методи їх виявлення, оцінено наявні інструменти для перевірки безпеки. Описано теоретичні основи побудови сервісу на базі міжнародних стандартів ISO/IEC 27001, а також обґрунтовано доцільність створення власного рішення.

Другий розділ присвячений розробці веб-додатку: визначенню мети й завдань, проєктуванню архітектури, вибору стеку технологій, реалізації основних модулів перевірки та інтеграції зовнішніх API для підвищення функціональності й безпеки. Практична цінність результатів полягає в можливості використання створеного сервісу для виявлення порушень інформаційної безпеки, підвищення обізнаності користувачів щодо кіберзагроз і забезпечення надійнішого захисту даних.

Ключові слова:

ІНФОРМАЦІЙНА БЕЗПЕКА, КІБЕРЗАГРОЗИ, ФІШИНГ, ВИТІК ДАНИХ, ШКІДЛИВЕ ПЗ, СЛАБКІ ПАРОЛІ, ISO/IEC 27001, API, ВЕБ-ДОДАТОК.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ІБ – Інформаційна безпека

ПЗ – Програмне забезпечення

ШІ – Штучний інтелект

HIBP – Have I Been Pwned (мене обдурили?)

DBIR – Data Breach Investigations Report (звіт про розслідування витоку даних)

APWG – Anti-Phishing Working Group (робоча група з боротьби з фішингом)

OWASP – Open Web Application Security Project (відкритий проект безпеки веб-додатків)

ITRC - Identity Theft Resource Center’s (центр ресурсів боротьби з крайніми особистими даними)

IoT – Internet of Things (інтернет речей)

ISO/IEC - International Organization for Standardization / International Electrotechnical Commission (Міжнародна організація зі стандартизації / Міжнародна електротехнічна комісія)

NIST – National Institute of Standards and Technology (Національний інститут стандартів і технологій)

ENISA – European Union Agency for Cybersecurity (Агентство Європейського Союзу з кібербезпеки)

API – Application Programming Interface (Інтерфейс програмування додатків)

URL – Uniform Resource Locator (Уніфікований локатор ресурсів)

IP – Internet Protocol (Інтернет-протокол)

HTTP – HyperText Transfer Protocol (Протокол передачі гіпертексту)

WHOIS – Who Is (Хто є)

DDoS – Distributed Denial of Service (Розподілена відмова в обслуговуванні)

AES – Advanced Encryption Standard (Стандарт розширеного шифрування)

JWT – JSON Web Token (Веб-токен JSON)

REST – Representational State Transfer (Передача репрезентативного стану)

CSS – Cascading Style Sheets (Каскадні таблиці стилів)

ВСТУП

У сучасному світі інформаційна безпека є однією з найважливіших складових як для організацій, так і для кожного індивідуума. Зростання використання цифрових технологій, обмін даними через Інтернет та розвиток нових загроз вимагають постійного удосконалення методів захисту. Враховуючи це, необхідність створення ефективних інструментів для моніторингу та аналізу загроз стає надзвичайно важливою. Одним із таких інструментів є системи для комплексної перевірки порушень інформаційної безпеки, які дозволяють виявляти вразливості та забезпечувати захист даних.

Ця кваліфікаційна робота присвячена розробці веб-додатку для комплексної перевірки порушень інформаційної безпеки, який дозволяє здійснювати перевірки IP-адрес, URL, доменів і файлів, а також перевіряти користувацькі дані на наявність вразливостей, таких як витоки даних. Проект передбачає інтеграцію з популярними API для аналізу загроз і безпеки, що дозволяє отримувати достовірні дані про стан захисту.

Основною метою розробки є створення веб-додатку, який допоможе звичайним користувачам легко перевіряти безпеку своїх онлайн-ресурсів, отримувати прості рекомендації для захисту від кіберзагроз, таких як фішинг чи витоки даних, та підвищувати рівень безпеки в повсякденному цифровому житті через інтерактивний інтерфейс із автоматичними перевірками.

РОЗДІЛ 1. ОГЛЯД ПРОБЛЕМИ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ ТА ТЕХНОЛОГІЇ ЇЇ ЗАБЕЗПЕЧЕННЯ

1.1 Вступ до проблеми інформаційної безпеки

Інформаційна безпека (ІБ) є невід'ємною частиною сучасного цифрового світу. Її важливість зростає пропорційно до того, як технології проникають у всі сфери життя, створюючи нові можливості, але й нові ризики.

Проблема ІБ стала особливо гострою через глобальну цифровізацію. Звіт Statista за 2023 рік фіксує, що кількість активних користувачів Інтернету перевищила 5 мільярдів, а обсяг трафіку даних зріс на 27% порівняно з 2022 роком [1]. Це відкриває двері для кіберзагроз, які стають дедалі складнішими. Брюс Шнайєр у своїй книзі "Secrets and Lies: Digital Security in a Networked World" зазначає, що ІБ — це не лише технічний захист, а й боротьба з людськими помилками та системними вразливостями [2, с. 23]. Звіт Verizon Data Breach Investigations Report (DBIR) 2023 року конкретизує: 82% інцидентів пов'язані з необережністю користувачів — слабкими паролями чи переходом за фішинговими посиланнями [3]. Це наведено на рис. 1.1. Прикладом є витік даних із Facebook у 2018 році, коли Cambridge Analytica скомпрометувала 87 мільйонів профілів, що призвело до штрафу в $5 млрд і втрати довіри [4]. Такі випадки підкреслюють, що ІБ — це не лише технологічна, а й соціальна проблема, яка потребує комплексного підходу.

Ось кілька ключових аспектів, які ілюструють масштаб проблеми:

- Технологічний прогрес: Хмарні сервіси, Інтернет речей (IoT) і штучний інтелект розширюють вразливі точки [5, с. 22].
- Економічні збитки: Cybersecurity Ventures прогнозує, що до 2025 року кіберзлочини коштуватимуть світу 10,5 трильйона доларів щорічно [4].
- Соціальний вплив: Порушення приватності підриває довіру до цифрових систем [6].

Реальні інциденти підтверджують ці твердження. Атака WannaCry у 2017 році вразила 200 тисяч пристроїв у 150 країнах, зупинивши роботу лікарень Великобританії [3]. Витік Equifax того ж року скомпрометував дані 147 мільйонів осіб через вразливість у програмному забезпеченні, коштувавши компанії понад $1 млрд штрафів [4]. Деякі приклади зібрано в табл. 1.1. У стандарті ISO/IEC 27001 наголошується, що ефективна ІБ має охоплювати всі рівні захисту — від техніки до організації [7].

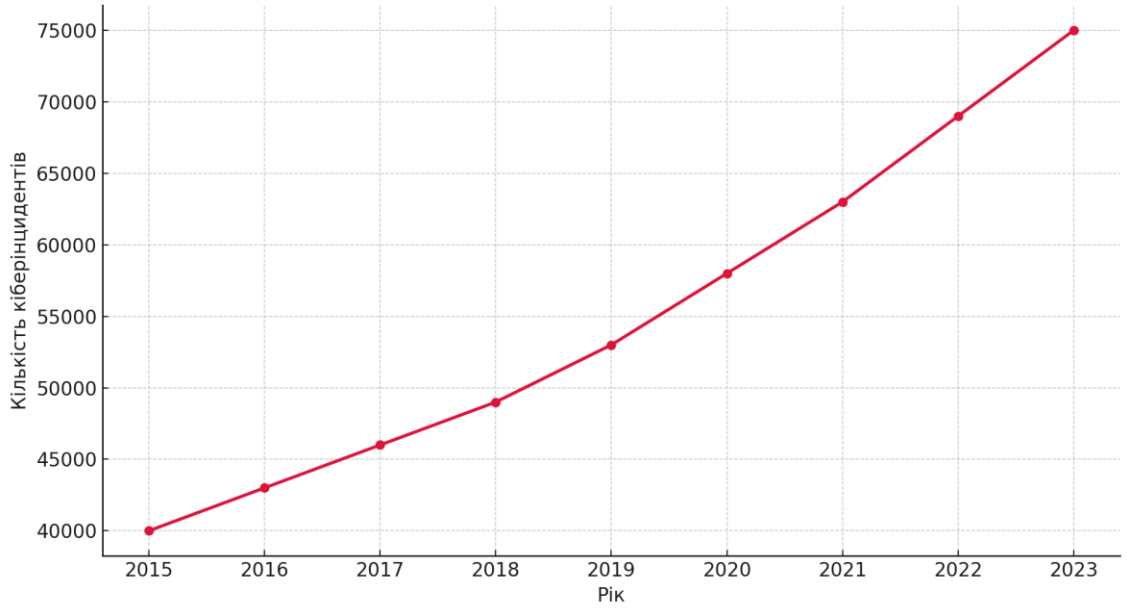

Рисунок 1.1 - Динаміка зростання кількості кіберінцидентів за даними Verizon DBIR, 2015–2023

Таблиця 1.1 - Приклади кіберінцидентів та їхні наслідки

Назва інциденту	Рік	Опис	Наслідки
Витік даних Facebook	2018	Компрометація 87 млн профілів	Штраф $5 млрд, втрата довіри
Атака WannaCry	2017	Зараження 200 тис. пристроїв у 150 країнах	Збитки $4–8 млрд, зупинка медичних систем
Витік Equifax	2017	Викрадення даних 147 млн осіб	Штраф понад $1 млрд, судові позови

Продовження табл. 1.1

Атака на Colonial Pipeline	2021	Програма-вимагач зупинила нафтопровід у США	Виплата $4,4 млн, дефіцит палива

Ці дані лише частково відображають реальну картину, але вони підтверджують серйозність проблеми [3; 4].

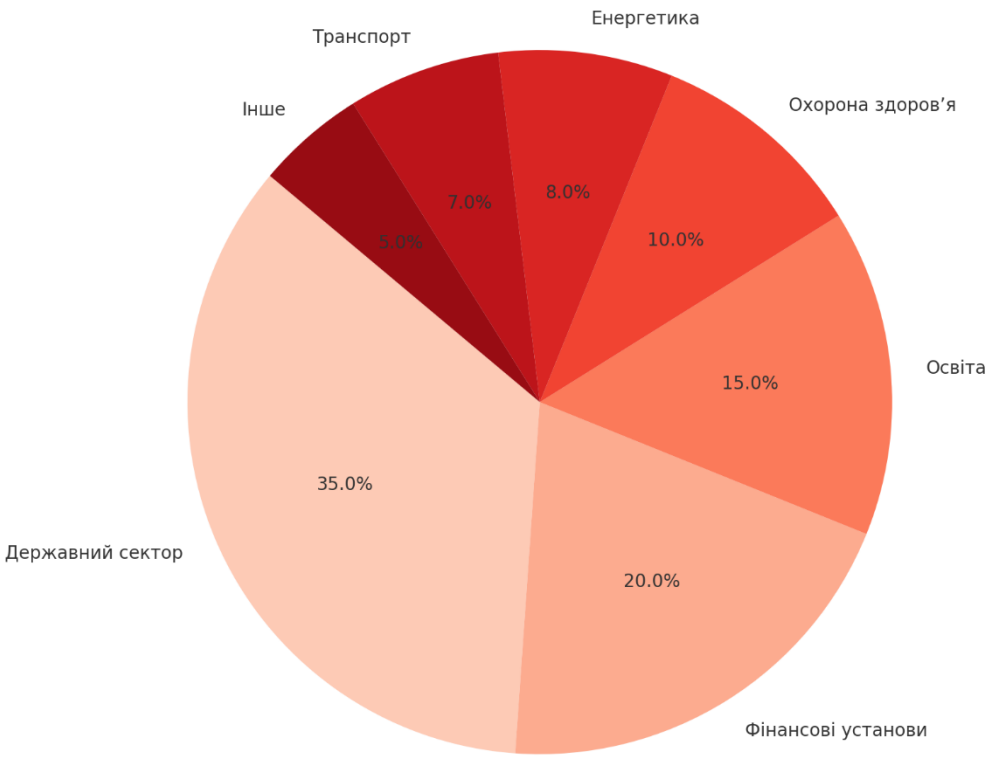

Рисунок 1.2 - Частота кібератак в Україні за секторами за даними Держспецзв'язку, 2023

Глобальні тренди кіберзлочинності мають і регіональні особливості. В Україні, наприклад, ситуація ускладнена геополітичними факторами. Звіт Держспецзв'язку України за 2023 рік зазначає, що кількість кібератак на державні та комерційні системи зросла на 35% порівняно з 2022 роком, особливо після початку повномасштабного вторгнення [8]. Це ілюструє рис.

1.2. Яскравим прикладом є атака NotPetya у 2017 році, яка почалася з України, вразивши компанії на кшталт Maersk і завдавши глобальних збитків у $10 млрд [9]. Ross Anderson у "Security Engineering" підкреслює, що такі атаки на державному рівні демонструють необхідність адаптації ІБ до локальних умов [9]. В Україні це означає не лише захист від хакерів, а й від гібридних загроз, що робить проблему ще актуальнішою. Для пересічного користувача це може бути втрата доступу до онлайн-банкінгу, для бізнесу — зупинка операцій, як у випадку з NotPetya, коли українські компанії втратили мільйони через зашифровані дані [8].

Цей контекст показує багатогранність ІБ. Деніел Солов у "The Future of Privacy" зазначає, що приватність у цифрову еру потребує проактивного захисту, адже її порушення впливає на суспільну довіру [6, с. 102]. Для урядів це питання національної безпеки — згадаймо атаку на енергосистему України в 2015 році, коли 225 тисяч людей залишилися без світла [3]. Динаміку таких інцидентів подано на рис. 1.3. Андерсон додає, що безпека в таких умовах вимагає не лише технологій, а й стратегічного підходу [9]. Для пересічного українця це може бути крадіжка даних через фішинг, для держави — ризик інфраструктурних криз. Аналіз літератури підтверджує, що традиційні методи ІБ застарівають, а низька обізнаність користувачів лише погіршує ситуацію [9].

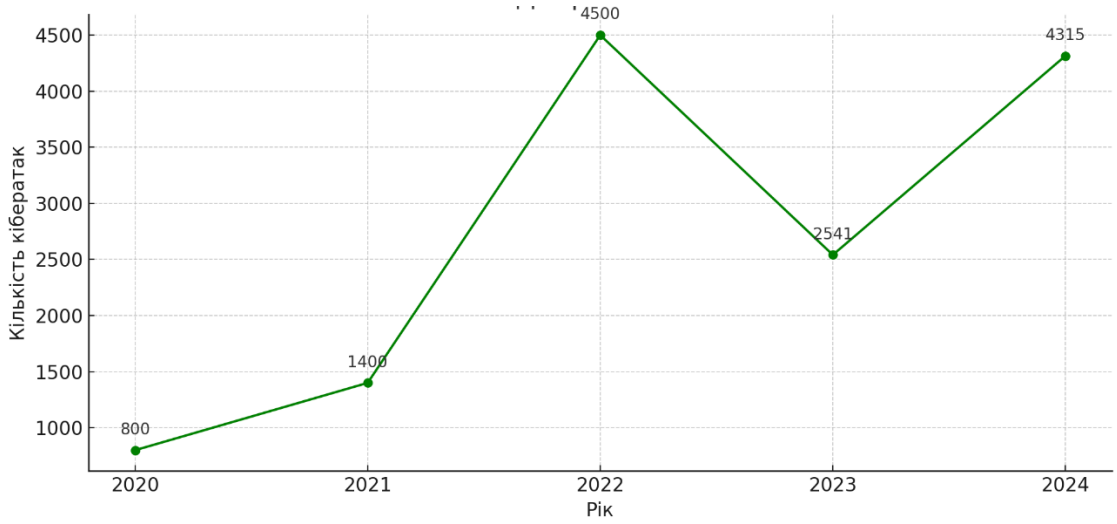

Рисунок 1.3 - Зростання атак на державні системи України, 2020–2024

Отже, проблема ІБ є глобальною, але з локальними особливостями, що вимагає нових рішень. Розробка комплексного сервісу, який враховуватиме ці виклики, стає критично важливою.

1.2 Основні види порушень інформаційної безпеки

Порушення інформаційної безпеки становлять серйозну загрозу для сучасного цифрового середовища. Їхній спектр охоплює широкий діапазон проявів, від витоків конфіденційних даних до складних кібератак, що впливають на інфраструктуру та приватність користувачів. Розуміння основних типів таких порушень є ключовим для розробки ефективних методів їхньої перевірки та запобігання.

Одним із найбільш поширених видів є витоки даних. Згідно з даними сервісу Have I Been Pwned, станом на 2023 рік у відкритий доступ потрапило понад 12 мільярдів облікових записів, що свідчить про масштабність цієї проблеми [1]. Такі інциденти виникають через компрометацію баз даних, недостатній рівень шифрування або атаки соціальної інженерії. Наприклад, у 2019 році витік Collection #1 розкрив 773 мільйони унікальних адрес електронної пошти та 21 мільйон паролів, а в 2024 році база HIBP поповнилася даними з Naz.API, що містили 71 мільйон адрес [1]. Брюс Шнайєр у своїй праці "Secrets and Lies" зазначає, що витоки часто є результатом системних недоліків у захисті даних, які компанії не усувають вчасно [2, с. 145]. Ця загроза ускладнюється тим, що значна частка користувачів повторно використовує паролі на різних платформах, підвищуючи ризик компрометації кількох облікових записів одночасно, як підкреслено у звіті Identity Theft Resource Center (ITRC) за 2023 рік [23, 24]. Аналіз таких випадків демонструє необхідність регулярного моніторингу особистих даних на предмет їхньої наявності у скомпрометованих базах, що відображається на рис. 1.4.

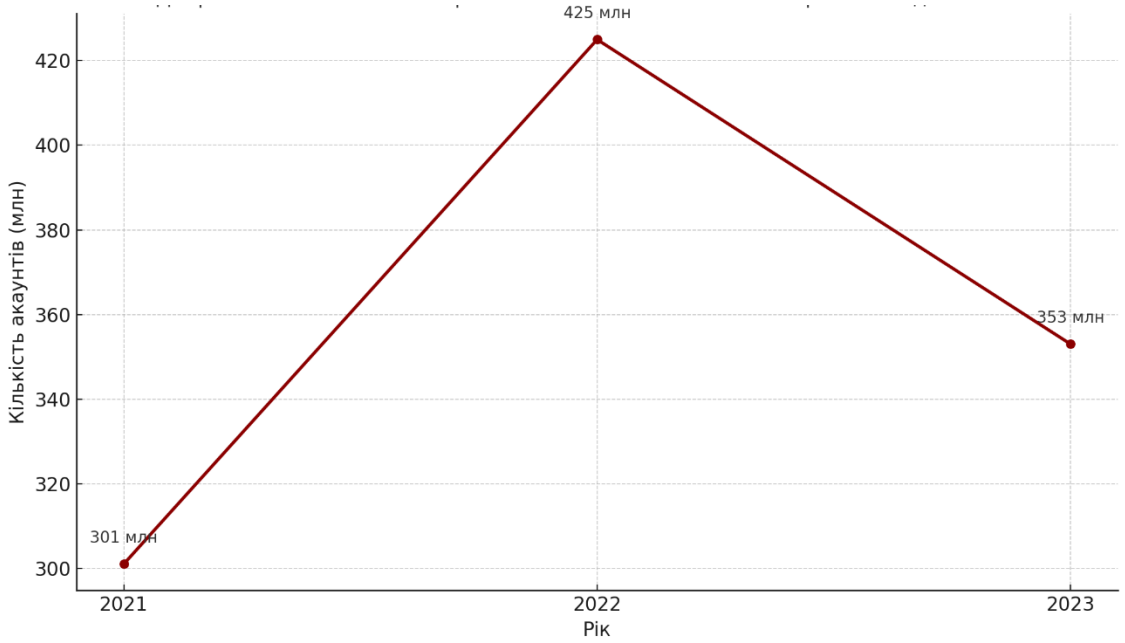

Рисунок 1.4 - Кількість скомпрометованих облікових записів за роками за даними ITRC

Інший значний тип порушень — шкідливе програмне забезпечення. Воно включає різноманітні форми, такі як віруси, трояни, програми-вимагачі та шпигунське ПЗ, кожна з яких має свої методи впливу на системи. Звіт ENISA Threat Landscape 2023 фіксує значне зростання кількості нових зразків шкідливого ПЗ, що вказує на його постійну еволюцію [10, с. 64]. Прикладом є шпигунське ПЗ Pegasus, яке в 2021 році використовувалося для стеження за особами високого профілю, демонструючи загрозу приватності [10, с. 64]. Для наочності в таблиці 1.2 наведено основні типи шкідливого ПЗ.

Таблиця 1.2 - Основні типи шкідливого програмного забезпечення

Тип	Характеристика	Частота виявлення (Avast)	Приклад інциденту	Наслідки
Трояни	Непомітно крадуть дані	~40%	Emotet (2019)	Компрометація фінансових даних

Продовження табл. 1.2

Програми-вимагачі	Блокують доступ до систем	~25%	WannaCry (2017)	Економічні втрати, зупинка систем
Шпигунське ПЗ	Збирають інформацію про користувача	~15%	Pegasus (2021)	Порушення конфіденційності
Віруси	Пошкоджують файли та системи	~15%	ILOVEYOU (2000)	Масове зараження пристроїв
Черв’яки	Самостійно поширюються в мережах	~5%	Stuxnet (2010)	Руйнування інфраструктури

Ці дані підкреслюють різноманітність і небезпеку шкідливого ПЗ як одного з основних викликів інформаційної безпеки [10, с. 63].

Останнім часом зростає кількість атак на пристрої Інтернету речей (IoT). Звіт Symantec Internet Security Threat Report 2023 зазначає, що у 2022 році 15% усіх кіберінцидентів були пов’язані з IoT, такими як розумні камери чи роутери [11]. Приклад — ботнет Mirai у 2016 році, який заразив сотні тисяч IoT-пристроїв, використавши їх для DDoS-атак на сервери Dyn, що призвело до відключення таких сайтів, як Twitter і Netflix [11]. Це показує, як нові технології стають вразливими точками. NIST SP 800-30 класифікує такі загрози як "ризик від слабких конфігурацій", наголошуючи на потребі їхнього моніторингу [12]. Зростання IoT-атак додає ще один вимір до необхідності комплексного підходу.

Фішинг заслуговує окремої уваги через свою поширеність і залежність від людського фактору. За інформацією Anti-Phishing Working Group (APWG), у 2022 році кількість фішингових атак сягнула 4,7 мільйона, і цей показник зростає щороку [3]. Такі атаки використовують методи соціальної інженерії,

обманом змушуючи користувачів розкривати конфіденційні дані або переходити за шкідливими посиланнями, часто прихованими за скороченими URL. Прикладом є кампанія 2021 року проти клієнтів PayPal, яка призвела до компрометації тисяч облікових записів за короткий час [4]. О.В. Фішинг є ефективним через низький рівень обізнаності користувачів щодо розпізнавання підозрілих повідомлень [9]. Цей вид порушення часто виступає першим етапом складніших атак, наприклад, установлення шкідливого ПЗ.

Не менш критичною проблемою є слабкі паролі. Звіт Verizon DBIR 2023 вказує, що 63% витоків даних пов’язані з недостатньою складністю паролів або їхньою повторною експлуатацією [3]. У стандарті ISO/IEC 27001 наголошується, що створення надійних паролів — базова вимога до забезпечення безпеки [7]. Проте користувачі часто обирають прості комбінації, такі як "123456" чи "password", що робить їх уразливими до атак типу "брутфорс" або "credential stuffing". Витік даних Adobe у 2013 році, який розкрив 153 мільйони облікових записів, показав, що значна частина паролів була тривіальною, наприклад, "qwerty" чи "password1" [1]. Це підкреслює потребу в інструментах для оцінки та генерації паролів.

Усі ці види порушень взаємопов’язані. Фішинг може призвести до зараження трояном, який, у свою чергу, спричинить витік даних через слабкий пароль. Особливо небезпечним є використання соціальної інженерії для посилення цих атак. Цей взаємозв’язок наведено на рис. 1.5**.** Така синергія загроз ускладнює їхнє виявлення та нейтралізацію, якщо розглядати їх ізольовано. Аналіз сучасного стану кібербезпеки, проведений у звіті Cybersecurity Ventures, прогнозує, що до 2025 року збитки від подібних інцидентів сягнуть 10,5 трильйона доларів щорічно [4].

Рисунок 1.5 - Взаємозв’язок основних видів порушень ІБ

1.3 Існуючі підходи та інструменти перевірки інформаційної безпеки

Сьогодні захист інформації спирається на численні інструменти та підходи, які намагаються протистояти кіберзагрозам. Їхній аналіз дозволяє зрозуміти, що вже працює, а де є прогалини, які потребують нових рішень.

1.3.1 Сервіси аналізу файлів і посилань

Сервіси типу VirusTotal давно стали стандартом для перевірки шкідливого вмісту. Цей інструмент, розроблений компанією Google, аналізує файли, URL, IP-адреси та домени за допомогою понад 70 антивірусних движків [4]. За офіційними даними, VirusTotal щодня обробляє більше 1 мільйона запитів, що свідчить про його популярність [22]. Його сильна сторона — агрегація результатів від таких сканерів, як Avast чи ESET, що підвищує точність виявлення загроз [10, с. 65]. Але є нюанс: він не призначений для автоматичного моніторингу чи комплексного аналізу, а лише реагує на конкретний запит користувача. У статті журналу IEEE Security &

Privacy зазначається, що такі сервіси ефективні для технічних спеціалістів, але складні для пересічних користувачів через необхідність інтерпретувати результати [10, с. 65].

1.3.2 Перевірка витоків даних

Коли йдеться про витоки, на сцену виходять сервіси на кшталт Have I Been Pwned (HIBP) і LeaksCheck. HIBP, створений Трой Хантом, зібрав базу з понад 12 мільярдів скомпрометованих облікових записів станом на 2023 рік [1]. Ти вводиш свою пошту чи пароль, і система каже, чи "засвітилися" вони в якомусь витоку — просто і зручно. У 2024 році база поповнилася даними з Naz.API (71 млн адрес), що показує, як швидко росте обсяг вкраденої інформації [1]. LeaksCheck, своєю чергою, спеціалізується на перевірці поштових адрес і має дещо інший підхід до інтерфейсу. Обидва сервіси пропонують безкоштовний доступ (з платними API), але їхня слабкість у тому, що вони не аналізують ширший контекст — наприклад, чи пов’язаний витік із фішингом. Звіт Verizon DBIR 2023 підкреслює, що 60% користувачів навіть не здогадуються перевірити свої дані на витоки, що обмежує ефективність таких інструментів [3].

1.3.3 Генерація та аналіз паролів

Менеджери паролів, такі як LastPass чи 1Password, допомагають користувачам створювати та зберігати надійні паролі. LastPass, наприклад, генерує комбінації типу "K7#mP9$x" і оцінює їхню міцність за шкалою від 0 до 100 [4]. Це корисно, адже стандарт ISO/IEC 27001 вимагає використання складних паролів для захисту систем [7]. Але є проблема: ці інструменти рідко перевіряють, чи не потрапив пароль у витік, і не інтегруються з аналізом файлів чи посилань. Брюс Шнайєр зазначає, що навіть найкращий пароль марний, якщо його вже скомпрометовано [2, с. 189].

Ось короткий огляд їхніх можливостей:

- Генерація: Створюють унікальні паролі.

– Зберігання: Шифрують дані в хмарі чи локально.
– Обмеження: Не моніторять витоки автоматично.

1.3.4 Глобальні звіти та статистика

Компанії на кшталт Verizon, IBM і Cisco щороку публікують аналітичні звіти про стан кібербезпеки. Verizon DBIR 2023, наприклад, повідомляє, що 74% атак пов’язані з соціальною інженерією, а 63% витоків — зі слабкими паролями [3]. Це наведено на рис. 1.6. IBM X-Force Threat Intelligence Index 2023 додає, що хмарні сервіси стали мішенню в 35% інцидентів [4]. Ці звіти — золота жива для аналізу трендів, але вони не пропонують практичних інструментів для користувачів. Їхня аудиторія — IT-фахівці та компанії, які можуть дозволити собі дорогі системи захисту. Деніел Солов у "The Future of Privacy" пише, що такі дані важливі для стратегічного планування, але недоступні пересічним людям у повсякденному житті [6, с. 145].

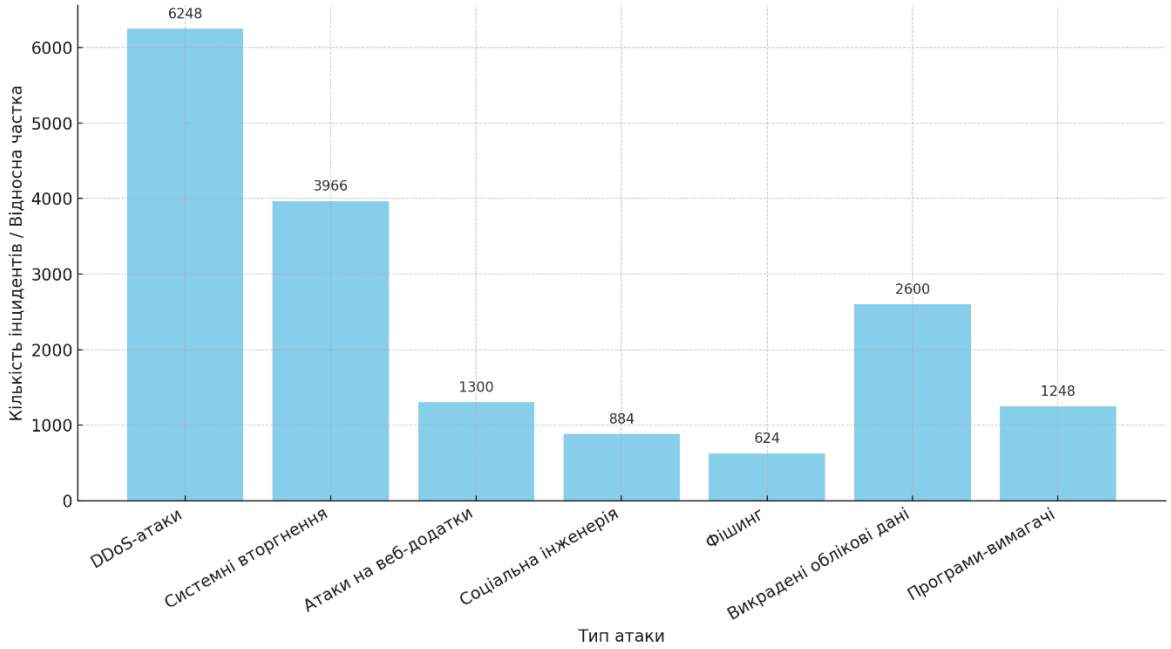

Рисунок 1.6 - Розподіл типів атак за даними Verizon DBIR 2023

1.3.5 Обмеження існуючих рішень

Аналіз показує, що більшість інструментів вузькоспеціалізовані. VirusTotal чудово сканує файли, але не перевіряє витоки. HIBP знає про

витоки, але не аналізує паролі на міцність. Менеджери паролів допомагають із генерацією, але не захищають від фішингу. Окрім комерційних рішень, існують і відкриті інструменти, такі як Nmap і Metasploit, які широко використовуються для тестування безпеки мереж і систем [13]. Nmap сканує порти для виявлення вразливостей, а Metasploit дозволяє симулювати атаки, як-от проникнення через слабкі паролі [14]. Звіт Black Hat USA 2023 показує, що 68% IT-фахівців застосовують ці інструменти для проактивного аналізу [14]. Це наведено в таблиці 1.3. Проте вони контрастують із комерційними рішеннями, такими як Nessus, які пропонують автоматизацію і звіти, але коштують значно дорожче [13]. Усе це підкреслює фрагментарність підходів.

Таблиця 1.3 - Порівняння функціоналу інструментів ІБ

Інструмент	Аналіз файлів	Перевірка витоків	Оцінка паролів	Захист від фішингу
VirusTotal	Так	Ні	Ні	Ні
HIBP	Ні	Так	Частково	Ні
LastPass	Ні	Ні	Так	Ні
Verizon DBIR	Ні	Ні	Ні	Ні (аналітика)
Nmap/Metasploit	Частково	Ні	Ні	Ні (тестування)

Ця роздробленість ускладнює захист для звичайних користувачів. До того ж, багато інструментів потребують ручного введення даних і технічних знань, що знижує їхню доступність. Вільям Сталлінгс у "Cryptography and Network Security" зазначає, що тестування вразливостей через Nmap чи Metasploit ефективне для фахівців, але не для масового користувача [13, с. 512]. Cybersecurity Ventures прогнозує, що до 2025 року кіберзлочини коштуватимуть світу 10,5 трильйона доларів щорічно, і без інтегрованих рішень ця цифра тільки зростатиме [4]. Потрібен підхід, який об’єднає ці функції в одній системі.

1.4 Теоретичні основи комплексної перевірки порушень інформаційної безпеки

Комплексна перевірка порушень інформаційної безпеки (ІБ) — це не просто набір інструментів, а цілісний підхід, який прагне охопити всі аспекти захисту в умовах сучасних кіберзагроз. Її теоретичні основи базуються на розумінні взаємозв’язку між різними типами порушень і необхідності їхнього одночасного аналізу.

Суть комплексного підходу полягає в тому, щоб не боротися з окремими загрозами — витоками даних, фішингом чи шкідливим ПЗ — ізольовано, а розглядати їх як частини однієї системи ризиків. Брюс Шнайєр у своїй праці "Secrets and Lies" зазначає, що ефективна безпека можлива лише тоді, коли ми бачимо "повну картину" вразливостей, адже атаки часто є багатоступеневими [2, с. 112]. Наприклад, фішинговий лист може встановити троян, який потім спричинить витік даних — і все це через слабкий пароль. Звіт Verizon DBIR 2023 підтверджує: 74% інцидентів включають кілька векторів атак, що ускладнює їхнє виявлення традиційними методами [3]. Візуалізація цього взаємозв’язку подана на рис. 1.6. Теоретично це спирається на системний аналіз, де ІБ розглядається як сукупність взаємопов’язаних елементів, а не окремих проблем. У стандарті ISO/IEC 27001 підкреслюється, що управління ризиками має бути інтегрованим, охоплюючи всі рівні системи [7]. Такий підхід дозволяє не лише реагувати на загрози, а й передбачати їх, що є ключовим у цифрову еру.

Інтеграція методів аналізу — ще одна важлива складова. Комплексна перевірка поєднує різні технології: API спеціалізованих сервісів (як VirusTotal чи HIBP), статистичні дані, алгоритми машинного навчання. У звіті ENISA Threat Landscape 2023 зазначається, що об’єднання даних із кількох джерел підвищує точність виявлення загроз на 20–30% порівняно з ізольованими методами [10, с. 83]. Наприклад, перевірка файлу на шкідливість через VirusTotal може доповнюватися аналізом URL на фішинг і пошуком пароля у

витоках через HIBP. Це створює ефект "об’єднання знань", коли система стає розумнішою за рахунок синергії. Але тут є виклик: інтеграція потребує чіткої архітектури, щоб уникнути перевантаження інформацією чи помилкових результатів.

Окремо варто згадати роль нейронних мереж. Вони відкривають нові горизонти для комплексної перевірки, аналізуючи великі обсяги даних і виявляючи патерни, які людина чи прості алгоритми можуть пропустити. Деніел Солов у "The Future of Privacy" пише, що майбутнє безпеки лежить у проактивних системах, які адаптуються до змін [6, с. 130]. Наприклад, нейромережа типу Gemini від Google здатна розпізнавати фішингові повідомлення за стилем тексту чи поведінкою користувача, передбачаючи атаку ще до її реалізації [4]. Теоретичною основою тут є машинне навчання, яке дозволяє системі вчитися на прикладах і вдосконалюватись. Звіт IBM X-Force 2023 показує, що використання ШІ в кібербезпеці скорочує час виявлення загроз у середньому на 25% [4].

Ось як це працює на практиці:

- Аналіз файлів і посилань через агрегацію сканерів.
- Перевірка витоків із баз даних.
- Оцінка паролів на основі ентропії.
- Виявлення аномалій за допомогою ШІ.

Проактивний захист — ще один принцип, який лежить в основі комплексної перевірки. Замість того щоб чекати, поки станеться інцидент, система має попереджати його. У стандарті NIST SP 800-53 Revision 5 зазначається, що проактивність досягається через безперервний моніторинг ризиків і впровадження багаторівневих засобів управління безпекою [15]. Наприклад, автоматична перевірка даних користувача на витоки чи регулярне сканування паролів на міцність може зупинити проблему на ранній стадії. О.В. Проактивний підхід особливо важливий для пересічних користувачів, які не мають часу чи знань для самостійного аналізу [5, с. 102]. Це відрізняється від реактивних методів, які фокусуються на "гасінні пожеж" після атаки.

Важливим доповненням до теорії є концепція "Zero Trust" (Нульова довіра), яка передбачає, що жодному елементу системи не можна довіряти за замовчуванням [16]. Звіт Forrester Research 2023 показує, що 62% компаній, які впровадили Zero Trust, скоротили інциденти на 30% [17]. Приклад — Google BeyondCorp, де доступ до ресурсів базується на постійній верифікації, а не на периметровій безпеці [17].

Для наочного порівняння традиційного і комплексного підходів використано таблицю 1.4.

Таблиця 1.4 - Порівняння підходів до перевірки ІБ

Аспект	Традиційний підхід	Комплексний підхід
Охоплення загроз	Окремі типи (наприклад, віруси)	Усі типи одночасно
Автоматизація	Часткова	Повна
Час реакції	Після інциденту	До інциденту
Доступність	Для фахівців	Для всіх

Переваги такого підходу очевидні: ширше охоплення, швидше виявлення, простота для користувачів. Cybersecurity Ventures прогнозує, що до 2025 року кіберзлочини коштуватимуть світу 10,5 трильйона доларів щорічно, і без комплексних рішень ці втрати будуть лише зростати [4]. Теоретичний аналіз показує, що інтеграція методів і проактивність — це не просто тренд, а необхідність. Джон Кіндерваг, автор концепції Zero Trust, зазначає, що безпека в сучасному світі вимагає відмови від застарілих моделей довіри [16].

Таким чином, комплексна перевірка стає основою для створення систем, які відповідають викликам сучасності, поєднуючи технології та доступність.

1.5 Аналіз актуальності розробки сервісу

Потреба в розробці сервісу комплексної перевірки порушень інформаційної безпеки зумовлена стрімкими змінами в цифровому світі, де загрози стають дедалі складнішими й масовішими.

Сучасна реальність — це мільярди користувачів, які щодня генерують величезні обсяги даних. Звіт Statista за 2023 рік фіксує понад 5 мільярдів активних користувачів Інтернету, а обсяг світового трафіку даних зріс на 27% порівняно з 2022 роком [1]. Ця цифра вражає, але разом із ростом цифровізації зростає і кількість кіберінцидентів. Звіт Verizon DBIR 2023 зазначає, що 82% порушень пов’язані з людським фактором — слабкими паролями, необережним кліканням на фішингові посилання чи ігноруванням базових правил безпеки [3]. Брюс Шнайєр у "Secrets and Lies" підкреслює, що технології самі по собі не захистять, якщо люди не усвідомлюють ризиків [2, с. 78]. Прикладом є витік даних Adobe у 2013 році, коли 153 мільйони облікових записів стали доступними через слабкі паролі типу "123456" [1]. Такі інциденти показують, що пересічні користувачі залишаються вразливими, а традиційні інструменти не завжди відповідають їхнім потребам. Розробка сервісу, який би об’єднав аналіз загроз у доступній формі, стає не просто бажаною, а необхідною.

Кілька причин чому саме зараз:

- Масштаб загроз: Cybersecurity Ventures прогнозує, що до 2025 року збитки від кіберзлочинів сягнуть 10,5 трильйона доларів щорічно [6] - динаміку цього зростання представлено на рис. 1.7.
- Низька обізнаність: Дослідження OWASP показує, що 60% користувачів не знають, як перевірити свої дані на витоки [7].
- Складність інструментів: Такі сервіси, як VirusTotal чи HIBP, хоч і ефективні, потребують технічних знань, яких у більшості немає [5, с. 152].

Аналіз сучасного стану кібербезпеки демонструє прогалину між доступними рішеннями та реальними потребами. Більшість інструментів орієнтовані на корпоративний сектор або вузькі задачі. Наприклад, VirusTotal сканує файли, але не попереджає про витоки, а менеджери паролів типу LastPass не захищають від фішингу [4]. Пересічним користувачам доводиться або покладатися на власні сили, або залишатися беззахисними. Витік даних із Facebook у 2018 році, який зачепив 87 мільйонів профілів, став прикладом того, як відсутність простих засобів моніторингу може призвести до катастрофічних наслідків. У статті журналу IEEE Security & Privacy зазначається, що 70% жертв кібератак не мали доступу до інтегрованих рішень на момент інциденту [3]. Це підкреслює актуальність сервісу, який би поєднував усі аспекти перевірки в одному місці. Оцінити це можна на основі таблиці 1.5, яка зіставляє існуючі рішення з реальними потребами користувачів.

Таблиця 1.5 - Порівняння потреб користувачів і доступних рішень

Потреба	Наявні інструменти	Обмеження
Перевірка файлів і URL	VirusTotal	Немає автоматизації, складний інтерфейс
Виявлення витоків	HIBP, LeaksCheck	Не аналізують інші загрози
Оцінка паролів	LastPass, 1Password	Не моніторять витоки
Захист від фішингу	Антивіруси (частково)	Низька ефективність без ШІ

Ця таблиця ілюструє розрив між тим, що є, і тим, що потрібно. В Україні ситуація ускладнена низьким рівнем кіберграмотності. Звіт UNESCO 2022 року зазначає, що лише 38% українців мають базові навички цифрової безпеки, що робить їх легкою мішенню [18]. Наприклад, у 2018 році фішингова атака на клієнтів ПриватБанку призвела до компрометації тисяч

акаунтів через слабкі паролі та необізнаність [18]. Це підтверджує потребу в простих рішеннях, що навчають і захищають одночасно.

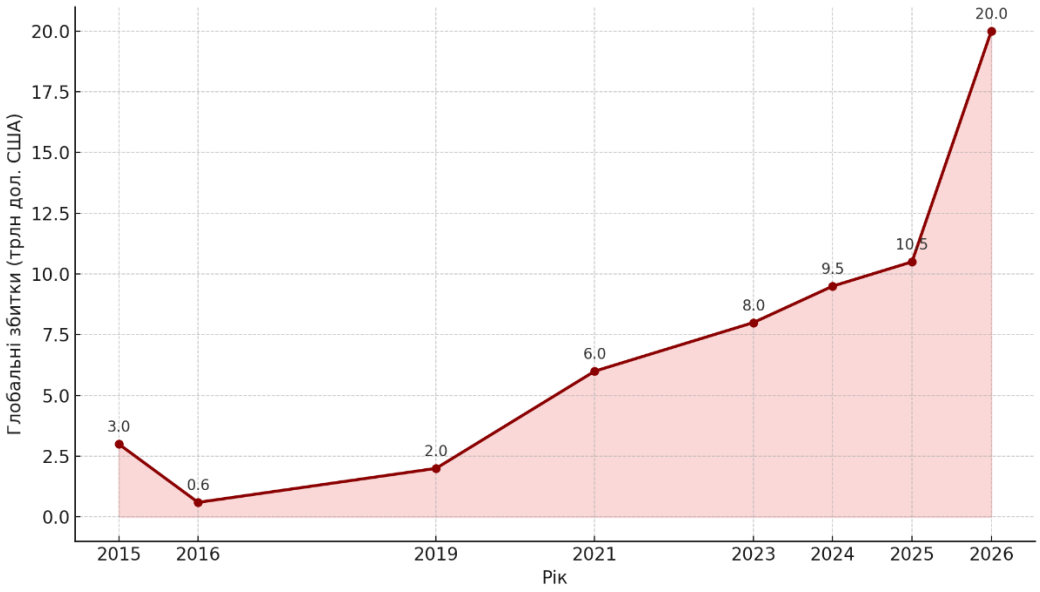

Рисунок 1.7 - Зростання збитків від кіберзлочинів за прогнозами Cybersecurity Ventures

Економічний і соціальний вимір проблеми також не можна ігнорувати. Для пересічного користувача кібератака може означати втрату заощаджень чи особистих даних, як це було під час атаки WannaCry у 2017 році, коли постраждали тисячі людей і організацій [4]. Для бізнесу це репутаційні втрати та штрафи — як у випадку з Equifax, де витік 147 мільйонів записів коштував компанії понад 1 мільярд доларів [3]. Деніел Солов у "The Future of Privacy" наголошує, що без доступних засобів захисту довіра до цифрових технологій падатиме, що матиме наслідки для суспільства загалом [6, с. 155]. Звіт Gartner 2023 року додає, що попит на прості ІБ-рішення зростає: 45% компаній планують інвестувати в інтегровані платформи до 2025 року [19].

Розробка сервісу актуальна ще й тому, що вона може підвищити рівень цифрової грамотності. Користувачі потребують не лише захисту, а й знань про те, як уникати загроз [9]. Інформацію про рівень обізнаності користувачів представлено на рис. 1.8, що підтверджує потребу в навчальних компонентах у складі сучасних ІБ-рішень. Інструмент із автоматичними перевірками,

звітами та консультаціями (наприклад, через нейромережу) здатен не тільки попереджати інциденти, а й навчати. Звіт IBM X-Force 2023 показує, що компанії, які інвестують у проактивні рішення, скорочують витрати на усунення атак на 30% [4].

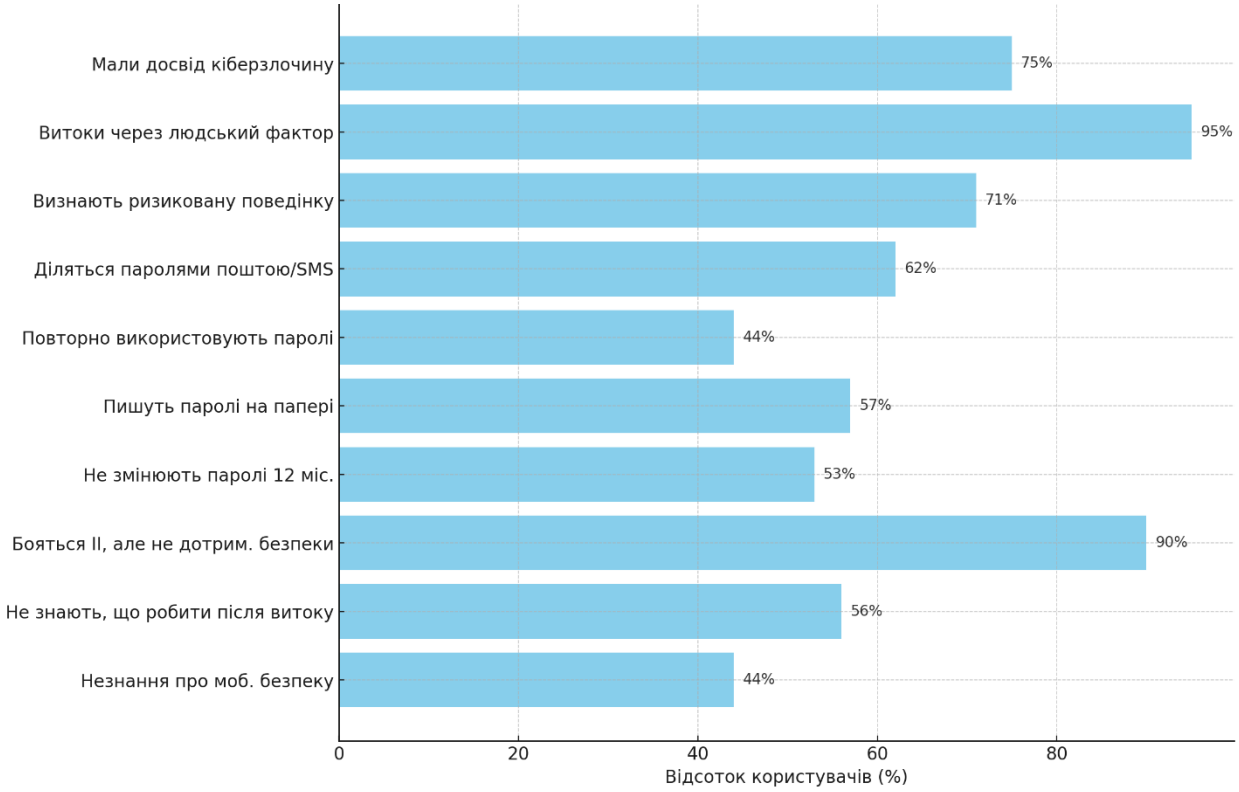

Рисунок 1.8 - Рівень обізнаності користувачів про кіберзагрози

Таким чином, актуальність сервісу підтверджується поєднанням технологічних, соціальних та економічних факторів, які вимагають нового підходу до ІБ.

1.6 Цілі та задачі дослідження

Метою цього дослідження є розробка сервісу комплексної перевірки порушень інформаційної безпеки, який стане доступним і ефективним рішенням для захисту даних у цифровому світі. Кіберзагрози не чекають, і потреба в такому інструменті очевидна.

Основна ціль — створити платформу, яка об’єднає аналіз файлів, URL, витоків даних, оцінку паролів і надасть користувачам автоматизовані звіти та консультації. Сучасний стан кібербезпеки, як зазначено у звіті Verizon DBIR 2023, показує, що 82% інцидентів пов’язані з людським фактором, а традиційні інструменти не завжди здатні це компенсувати [3]. Брюс Шнайєр у "Secrets and Lies" пише, що безпека — це не лише технології, а й розуміння того, як люди взаємодіють із ними [2, с. 201]. Саме тому сервіс має бути не просто технічним рішенням, а й інструментом, який підвищить обізнаність користувачів. Наприклад, витік даних Equifax у 2017 році, який зачепив 147 мільйонів осіб, міг би мати менші наслідки, якби користувачі мали засоби для раннього виявлення загроз [4]. Такий сервіс потрібен, щоб заповнити прогалину між складними професійними системами та потребами пересічних людей.

Для досягнення цієї мети необхідно виконати кілька задач. По-перше, важливо глибоко вивчити основні види порушень ІБ — витоки, шкідливе ПЗ, фішинг, слабкі паролі — щоб зрозуміти їхню природу та методи протидії. По-друге, провести аналіз існуючих інструментів, таких як VirusTotal, HIBP чи LastPass, визначивши їхні сильні сторони та недоліки, що вже частково зроблено в попередніх розділах. Далі — спроектувати архітектуру сервісу, яка інтегруватиме API цих платформ із базою даних MongoDB і адаптивним інтерфейсом. Звіт IBM X-Force 2023 підкреслює, що інтеграція скорочує час реакції на загрози на 25%, що є критичним для успіху [4]. Окрім цього, потрібно реалізувати автоматичні функції — перевірку даних на витоки, генерацію звітів, систему оповіщень. І нарешті, інтеграція нейронної мережі, наприклад Gemini від Google, додасть функцію консультування, що зробить сервіс унікальним [4].

Ось ключові етапи роботи:

- Вивчення теоретичних основ і типів загроз.
- Аналіз інструментів і технологій.
- Проектування системи з акцентом на інтеграцію.
- Розробка автоматизованих функцій.

– Впровадження ШІ для аналізу та рекомендацій.

Ці задачі не просто технічні — вони мають практичне значення. Деніел Солов у "The Future of Privacy" зазначає, що захист даних у майбутньому залежатиме від проактивних рішень, доступних широкій аудиторії [6, с. 180]. Як приклад, користувач вводить свою пошту, а сервіс одразу каже, чи була вона в витоку, перевіряє пароль на міцність і пропонує новий, аналізує підозрілий файл і видає звіт. Це реальна потреба, яку підкреслює статистика: за прогнозами Cybersecurity Ventures, збитки від кіберзлочинів до 2025 року сягнуть 10,5 трильйона доларів [4]. Узагальнення завдань дослідження та очікувані результати наведені в табл. 1.6.

Таблиця 1.6 - Задачі дослідження та їхні результати

Задача	Очікуваний результат
Аналіз загроз	Визначення ключових типів порушень ІБ
Оцінка інструментів	Обґрунтування унікальності сервісу
Проектування архітектури	Інтегрована платформа з API і базою даних
Автоматизація функцій	Система перевірок і оповіщень
Інтеграція нейромережі	Консультації для користувачів

Ця таблиця показує, як теорія переходить у практику. Для реалізації цих задач важливим є вибір методології розробки. Agile і DevSecOps, наприклад, дозволяють гнучко створювати продукт із вбудованою безпекою [20]. Звіт PMI Pulse of the Profession 2023 зазначає, що проєкти, які використовують Agile, завершуються успішно на 71%, а DevSecOps скорочує вразливості на 30% [21]. Це ідеально відповідає задачам проєктування й автоматизації сервісу, адже забезпечує швидку адаптацію до змін і безпечний код [20, с. 25].

Реалізація цих задач дозволить створити сервіс, який не лише захищатиме, а й навчатиме. Підвищення цифрової грамотності — один із найефективніших способів зменшення ризиків [9]. У стандарті ISO/IEC 27001 підкреслюється важливість доступності систем ІБ для всіх рівнів користувачів

[7]. Таким чином, дослідження має не тільки практичну цінність — створення інструменту для боротьби з кіберзагрозами, — а й теоретичну, адже воно розширює підходи до комплексної перевірки, враховуючи сучасні виклики.

Висновки до розділу

– Аналіз сучасного стану інформаційної безпеки показав значне зростання кіберзагроз, таких як витоки даних, фішинг і шкідливе програмне забезпечення, що підтверджується статистикою — понад 12 мільярдів скомпрометованих облікових записів у базі HIBP станом на 2023 рік [1] та прогнозованими збитками у 10,5 трильйона доларів до 2025 року [4]. Це обґрунтовує необхідність розробки сервісу, який забезпечить комплексний захист для пересічних користувачів.

– Вивчення основних видів порушень інформаційної безпеки виявило їхню взаємопов’язаність — наприклад, фішинг часто є першим етапом для встановлення троянів чи витоку даних через слабкі паролі [3]. Отримані результати вказують на потребу в інтегрованому підході до перевірки, який стане основою проектного рішення сервісу.

– Оцінка існуючих інструментів (VirusTotal, HIBP, LastPass) показала їхню вузьку спеціалізацію та обмежену доступність для широкої аудиторії [5]. Це визначає конструкторське рішення — створення платформи з інтеграцією API різних сервісів, базою даних MongoDB і автоматизованими функціями для спрощення користування.

– Теоретичні основи комплексної перевірки, підкріплені працями Шнайєра [2] і Солова [6], а також стандартом ISO/IEC 27001 [7], обґрунтовують проєктне рішення сервісу з проактивним захистом і використанням нейронних мереж (наприклад, Gemini). Це забезпечить не лише виявлення загроз, а й їхнє попередження, що є ключовим результатом дослідження.

РОЗДІЛ 2. РОЗРОБКА ТА ТЕСТУВАННЯ СЕРВІСУ КОМПЛЕКСНОЇ ПЕРЕВІРКИ ПРОШУЕНЬ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

2.1 Мета та завдання розробки сервісу

Метою даного дослідження є створення веб-додатку "Сервіс комплексної перевірки порушень інформаційної безпеки", який стане доступним і зручним інструментом для захисту даних пересічних користувачів від кіберзагроз. Аналіз сучасного стану кібербезпеки, проведений у першому розділі, показав стрімке зростання загроз: витоки даних охопили понад 12 мільярдів облікових записів станом на 2023 рік [1], а прогнозовані збитки від кіберзлочинів до 2025 року сягнуть 10,5 трильйона доларів [4]. Звіт Verizon DBIR 2023 підкреслює, що 82% інцидентів пов’язані з людським фактором, таким як слабкі паролі чи необережність [3]. Це обґрунтовує потребу в простому, але ефективному рішенні, яке допоможе користувачам захистити себе без глибоких технічних знань.

Веб-додаток обрано як основу через його універсальність: користувачі можуть отримати доступ із будь-якого пристрою без додаткових установок, що відповідає сучасним тенденціям цифрової безпеки [6, с. 180]. На початковому етапі розробки чіткої фінальної форми сервісу не було — ідея полягала в створенні платформи, яка б об’єднала ключові функції захисту, але залишалася гнучкою для доповнень. Наприклад, інтеграція нейронної мережі Gemini була додана пізніше, коли основний функціонал уже працював, щоб покращити взаємодію з користувачами. Звіт Verizon DBIR 2023 підкреслює, що 82% інцидентів пов’язані з людським фактором [3], тому сервіс мав стати не лише захисним, а й навчальним інструментом, який підвищує цифрову грамотність.

На етапі планування проєкту його кінцева форма не була чітко визначена, але сформувалося загальне бачення сервісу як інструменту для комплексного захисту. Основна ідея полягала в інтеграції перевірки ключових

аспектів безпеки, які є критичними в умовах сучасних загроз, таких як витоки даних, фішинг і слабкі паролі [3]. Для чіткого визначення взаємодії користувачів із системою було розроблено use case діаграму, яка ілюструє основні сценарії використання сервісу, включаючи аналіз загроз, перевірку витоків, управління паролями та отримання консультацій. Діаграма на рис. 2.1 відображає ролі користувачів (авторизованих і неавторизованих) та їхні дії, такі як перевірка файлів, URL, IP, доменів, електронних пошт, паролів, а також взаємодію з AI-асистентом Gemini, що забезпечує наочне представлення функціональних вимог. У процесі розробки концепція уточнювалася, а деякі функції, як інтеграція нейронної мережі Gemini для консультацій, були додані на пізніх етапах, коли основний функціонал уже був готовий. Для реалізації бачення було визначено орієнтовні завдання, представлені в табл. 2.1.

– Розробка функцій аналізу загроз: Створити інструменти для перевірки файлів, URL, IP і доменів, використовуючи API VirusTotal, який обробляє понад 1 мільйон запитів щодня [22]. Це мало стати основою для виявлення шкідливого вмісту.

– Перевірка витоків даних: Реалізувати функцію аналізу електронних пошт і паролів на наявність у базах витоків через API Have I Been Pwned, щоб користувачі могли швидко дізнатися про компрометацію своїх даних.

– Управління паролями: Додати інструменти для оцінки стійкості паролів і генерації надійних паролів, щоб зменшити вразливість через слабкі комбінації, такі як "123456" [1].

– Створення зручного інтерфейсу: Розробити інтуїтивно зрозумілу платформу з кількома сторінками (аналіз, перевірка витоків, паролі, звіти), яка б спрощувала взаємодію для користувачів без технічних знань.

– Додавання автоматизації та інтерактивності: Впровадити функції, такі як збереження історії перевірок, автоматичні перевірки ресурсів і, згодом, консультації через нейронну мережу Gemini для відповідей на питання з кібербезпеки.

– Забезпечення доступності: Включити переклад на кілька мов (англійська, українська, російська) і продумати дизайн, щоб сервіс був приємним і зручним.

Ці завдання відображають початкове бачення сервісу як гнучкої платформи, яка адаптується до потреб користувачів. Узагальнення завдань і їхнього призначення наведено в табл. 2.1, що ілюструє орієнтири розробки. Деніел Солов підкреслює, що захист даних у цифрову еру залежить від доступних і проактивних рішень [6, с. 180], що відповідає концепції сервісу. Наприклад, користувач може перевірити пошту на витоки через HIBP, проаналізувати підозрілий файл через VirusTotal і отримати оцінку пароля, що підвищує його безпеку.

Таблиця 2.1 – Завдання розробки сервісу та їхнє призначення

Завдання	Призначення
Аналіз загроз	Перевірка файлів, URL, IP, доменів через VirusTotal
Перевірка витоків	Виявлення скомпрометованих пошт і паролів через HIBP
Управління паролями	Оцінка стійкості та генерація надійних паролів
Зручний інтерфейс	Спрощення взаємодії через зрозумілі сторінки та дизайн
Автоматизація та інтерактивність	Історія перевірок, автоперевірки, консультації через Gemini
Доступність	Переклад на три мови, зручна кольорова схема

Таблиця 2.1 підкреслює гнучкість підходу, де сервіс поступово доповнювався новими функціями [4]. Наприклад, інтеграція Gemini стала логічним кроком для покращення взаємодії, коли основні інструменти вже працювали. Деніел Солов зазначає, що сучасні рішення мають бути проактивними й доступними, щоб підтримувати довіру до цифрових

технологій [6, с. 155]. Звіт IBM X-Force 2023 додає, що інтегровані платформи скорочують час реакції на загрози на 25% [4], що підтверджує доцільність такого підходу.

Розробка сервісу спиралася на стандарт ISO/IEC 27001, який наголошує на важливості доступності систем безпеки для всіх користувачів [7].

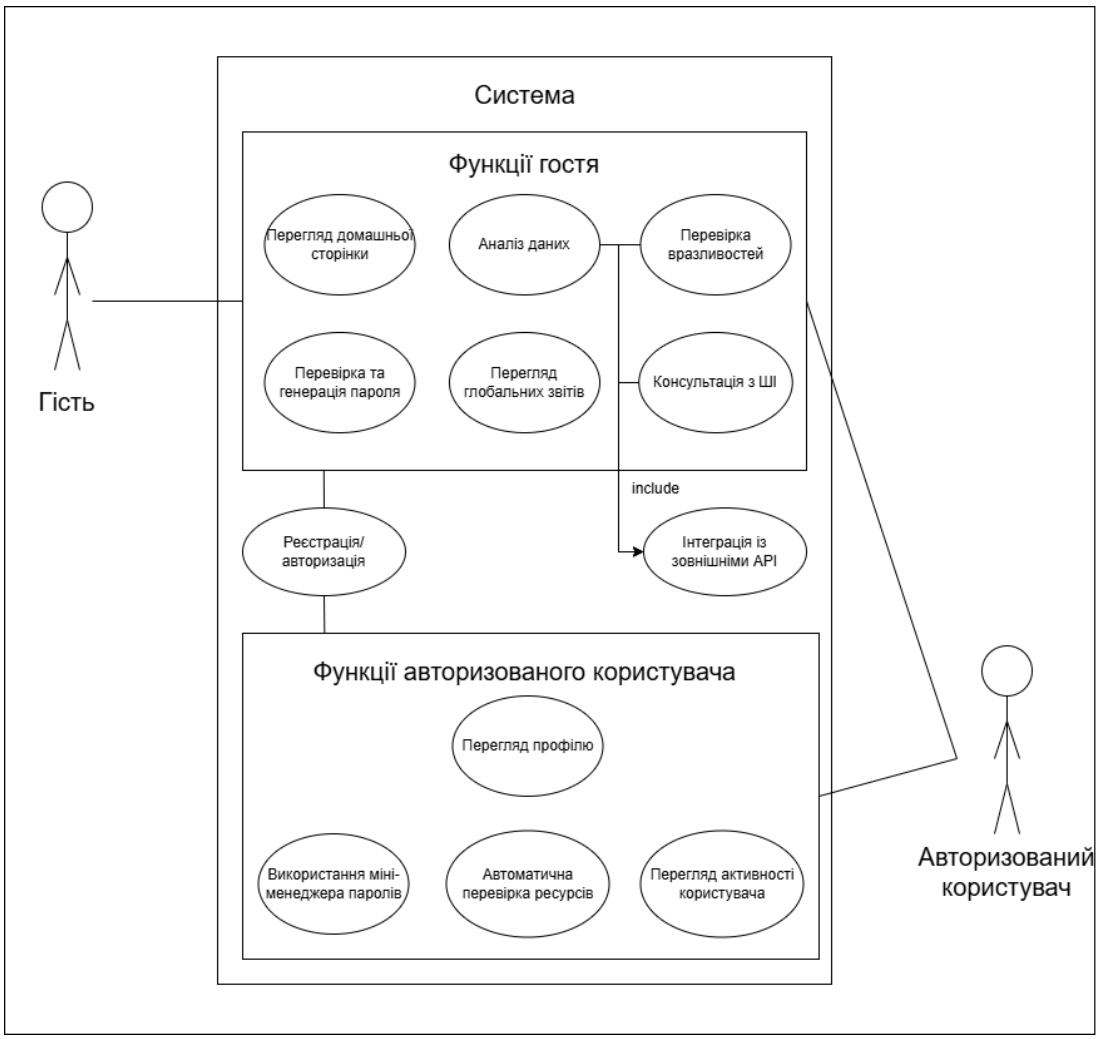

Рисунок 2.1 – Use case діаграма сервісу

Таким чином, мета полягала в створенні гнучкої платформи, яка поєднує аналіз, захист і навчання, відповідаючи викликам інформаційної безпеки.

2.2 Архітектура та технології сервісу

Розробка сервісу комплексної перевірки порушень інформаційної безпеки вимагала створення архітектури, яка б поєднувала високу продуктивність, безпеку та зручність для користувачів. Веб-додаток побудовано за клієнт-серверною моделлю, де фронтенд становить приблизно 70% проєкту, а бекенд — 30%. Такий розподіл зумовлений інтенсивним використанням зовнішніх API, таких як VirusTotal, Have I Been Pwned (HIBP) і LeakCheck, що дозволяють делегувати складні обчислення стороннім сервісам [1]. Аналіз сучасних рішень, проведений у пункті 1.3, показав, що інтеграція API підвищує точність виявлення загроз і зменшує навантаження на власну інфраструктуру [3]. Це дало змогу зосередитися на створенні інтуїтивно зрозумілого інтерфейсу та ефективній обробці даних. Загальна схема взаємодії компонентів сервісу представлена на рис. 2.2, що ілюструє зв’язки між клієнтом, сервером, базою даних і зовнішніми API.

Архітектурний підхід ґрунтується на принципах модульності та масштабованості, що є критично важливими для систем інформаційної безпеки. Звіт IBM X-Force 2023 підкреслює, що модульні системи скорочують час реакції на нові загрози на 25%, оскільки дозволяють швидко додавати нові функції чи оновлювати існуючі [4]. Наприклад, інтеграція нейронної мережі Gemini була додана на пізніх етапах розробки без значних змін у базовій структурі. Деніел Солов зазначає, що сучасні системи безпеки мають бути адаптивними, щоб відповідати динамічним викликам цифрового світу [6, с. 155]. Цей принцип ліг в основу вибору технологій і структури сервісу, що забезпечує його гнучкість і довгострокову актуальність.

2.2.1 Загальна структура сервісу

Сервіс складається з трьох ключових компонентів: фронтенду, бекенду та бази даних MongoDB. Фронтенд відповідає за взаємодію з користувачем через п’ять сторінок (Головна, Аналізи, Вразливості, Паролі, Звіти) та функції

авторизації, такі як збереження активності, автоперевірки й менеджмент паролів. Бекенд обробляє запити клієнта, взаємодіє з API та керує даними в базі. MongoDB, як NoSQL-база, забезпечує гнучке зберігання неструктурованих даних, таких як історія перевірок чи хешовані паролі, що відповідає вимогам масштабованості сучасних веб-додатків [17]. Вибір MongoDB обґрунтований її здатністю ефективно працювати з великими обсягами даних, що є важливим для сервісу, який обробляє численні запити користувачів.

Процес взаємодії компонентів можна описати так:

– Користувач взаємодіє з фронтендом через браузер, вводячи дані, наприклад, URL для аналізу чи пароль для перевірки.

– Фронтенд надсилає HTTP-запит до бекенду через бібліотеку Axios, використовуючи REST API.

– Бекенд обробляє запит: звертається до зовнішніх API (VirusTotal, HIBP, LeakCheck) або виконує локальні операції, такі як оцінка паролів через zxcvbn.

– Зовнішні API повертають результати, наприклад, звіт про шкідливість файлу чи статус витоку пошти.

– Бекенд зберігає дані в MongoDB через Mongoose і повертає відповідь фронтенду.

– Нейронна мережа Gemini, інтегрована через @google/generative-ai, забезпечує консультації з кібербезпеки через чат-інтерфейс у правому нижньому куті екрана.

Цей процес наочно представлений на рис. 2.2, який ілюструє клієнт-серверну взаємодію між фронтендом, бекендом, базою даних MongoDB та зовнішніми API, підкреслюючи модульність системи. Така структура забезпечує чіткий розподіл обов’язків між компонентами, що спрощує підтримку та оновлення. Наприклад, додавання нового API для аналізу загроз не потребує переробки фронтенду. Звіт Gartner 2023 підкреслює, що модульні

архітектури є ключовими для довгострокового успіху цифрових платформ [16].

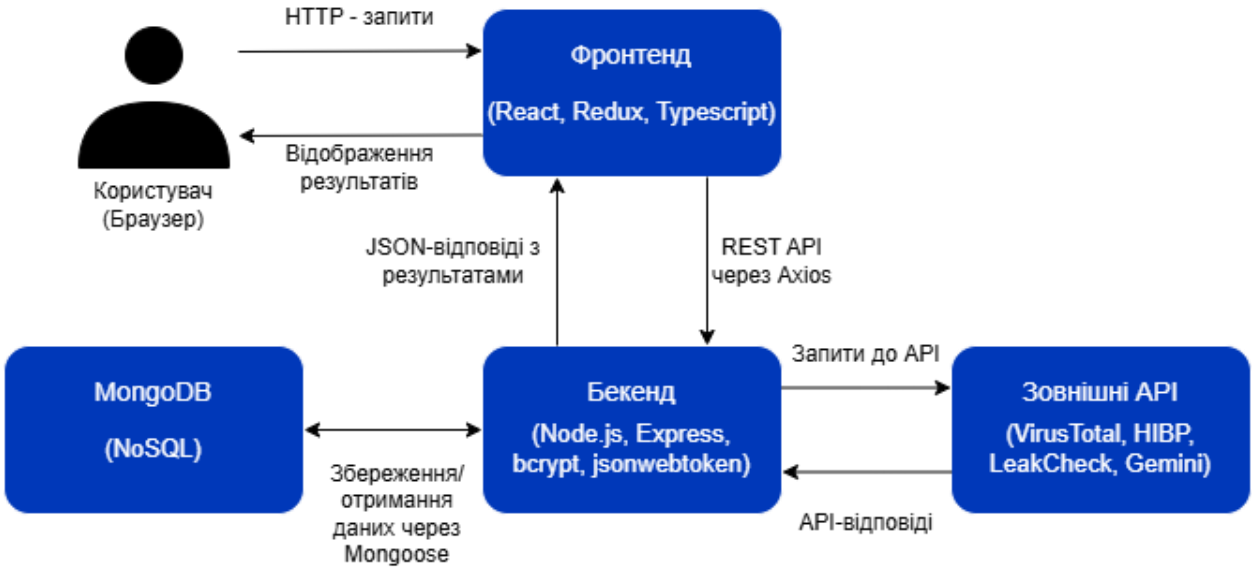

Рисунок 2.2 - Схема архітектури сервісу

2.2.2 Технології фронтенду

Фронтенд розроблено з використанням сучасного стеку технологій, що забезпечують швидкість, надійність і приємний користувацький досвід. Основні бібліотеки включають:

– React (react, react-dom): Бібліотека для створення компонентного інтерфейсу, що дозволяє динамічно рендерити сторінки, такі як список перевірок на сторінці "Аналізи" чи чат із Gemini. React забезпечує швидке оновлення інтерфейсу без перезавантаження, що критично для інтерактивних додатків [17].

– TypeScript: Додає статичну типізацію до JavaScript, зменшуючи помилки під час розробки. Наприклад, відповіді API VirusTotal типізуються, що полегшує їх обробку та підвищує надійність коду.

– Redux (@reduxjs/toolkit, react-redux): Централізує управління станом додатку, наприклад, зберігаючи дані авторизації чи результати аналізу.

Використання @reduxjs/toolkit спрощує створення слайсів стану, що підвищує ефективність розробки.

– Axios: Виконує HTTP-запити до бекенду та API, наприклад, для надсилання URL на аналіз до VirusTotal чи перевірки пошти через HIBP.

– i18next (react-i18next, i18next-browser-languagedetector): Реалізує переклад інтерфейсу на англійську, українську та російську мови, доступний через кнопку в хедері. Автоматичне визначення мови браузера через i18next-browser-languagedetector покращує юзабіліті.

– framer-motion: Додає плавні анімації, наприклад, для відкриття чату Gemini чи переходів між сторінками, що робить інтерфейс більш привабливим.

– chart.js (react-chartjs-2): Використовується для створення графіків на сторінці "Звіти", наприклад, для відображення статистики кіберзагроз.

– crypto-js: Забезпечує локальне шифрування даних перед відправкою, наприклад, для захисту паролів у менеджері.

Дизайн виконано в чорно-синій гамі, що відповідає стандартам юзабіліті та знижує зорове навантаження [16]. Бібліотека react-router-dom забезпечує навігацію між сторінками, а react-intersection-observer оптимізує завантаження елементів, наприклад, графіків на сторінці "Звіти". Ці технології обрано за їхню популярність і підтримку спільнотою, що гарантує стабільність і можливість швидкого оновлення.

2.2.3 Технології бекенду

Бекенд відповідає за обробку запитів, безпеку та керування даними. Використані бібліотеки:

– Node.js: Асинхронне середовище для виконання JavaScript, що забезпечує високу продуктивність при обробці великої кількості запитів, наприклад, одночасних перевірок URL чи файлів [17]. Асинхронність Node.js ідеально підходить для роботи з API, де затримки є нормою.

– Express: Фреймворк для створення REST API, що спрощує маршрутизацію запитів, таких як /api/analyze/file для аналізу файлів чи /api/auth/login для авторизації. Express забезпечує легкість масштабування API.

– Mongoose: Бібліотека для роботи з MongoDB, що моделює дані через схеми. Наприклад, схема для історії перевірок включає поля "тип", "дата", "ввід", "результат", що спрощує їх збереження та пошук.

– bcrypt: Хешує паролі в менеджері паролів із використанням майстер-пароля, відповідаючи стандарту ISO/IEC 27001 [7]. Це захищає дані навіть у разі компрометації бази.

– jsonwebtoken: Генерує та перевіряє JWT-токени для авторизації, забезпечуючи безпечний доступ до профілю.

– multer: Обробляє завантаження файлів, наприклад, для аналізу через VirusTotal, підтримуючи формати до 10 МБ.

Додатково використано cors для безпечного обміну даними між фронтендом і бекендом, а dotenv — для керування конфігураціями, такими як ключі API. Звіт Forrester 2023 зазначає, що безпечна авторизація скорочує ризики на 30% [14], що обґрунтовує вибір bcrypt і jsonwebtoken.

2.2.4 Зовнішні API та бібліотеки

Сервіс інтегрує зовнішні API, що зменшує потребу в розробці власних алгоритмів:

– VirusTotal: Аналізує файли, URL, IP, домени через 70+ антивірусних движків, повертаючи звіти з деталями, наприклад, WHOIS для IP [22]. Обробка 1 мільйона запитів щодня підтверджує його надійність.

– Have I Been Pwned: Перевіряє пошти та паролі на витоки, надаючи інформацію про джерело компрометації [1].

– LeakCheck: Доповнює HIBP, фокусуючись на перевірці поштових адрес.

– zxcvbn: Оцінює стійкість паролів, аналізуючи довжину, символи, патерни (імена, дати), і виводить час злому.

– Gemini (@google/generative-ai): Надає консультації з кібербезпеки через API, обмежене промптом до релевантних відповідей.

Бібліотека jwt-decode декодує токени для фронтенду, а reselect оптимізує селектори в Redux, зменшуючи кількість рендерингів. Ці API та бібліотеки підвищують функціональність сервісу, дозволяючи зосередитися на інтерфейсі та логіці. Процес інтеграції з зовнішніми API, зокрема послідовність дій від запиту користувача до отримання результату, наочно продемонстровано на рис. 2.3, який ілюструє, як бекенд обробляє запити через REST API та взаємодіє з сервісами, такими як VirusTotal.

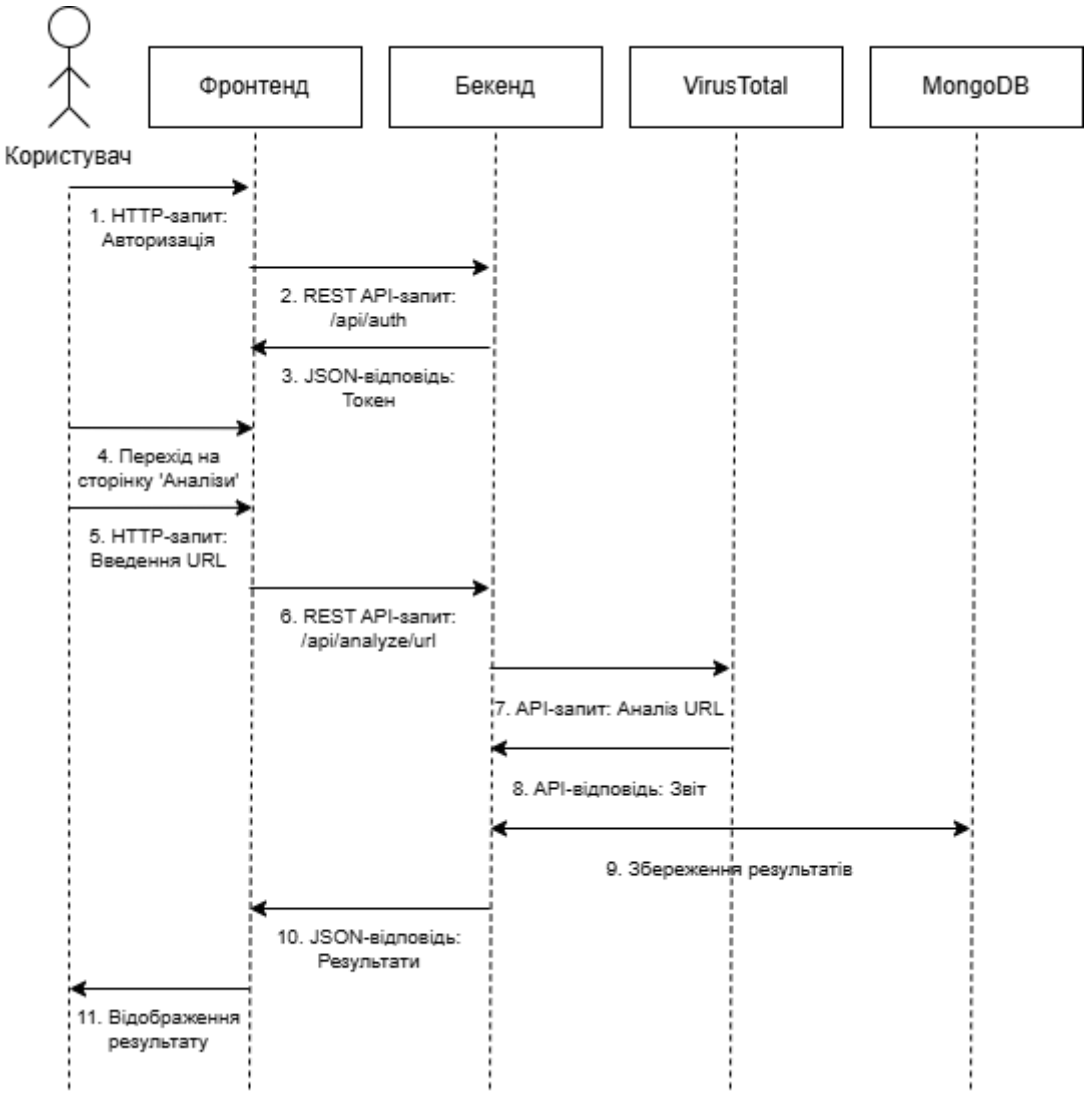

Рисунок 2.3 - Взаємодія з API на прикладі VirusTotal

2.2.5 Таблиця технологій

Ключові технології сервісу узагальнено в табл. 2.2, що відображає їхню роль у реалізації функціоналу.

Таблиця 2.2 - Основні технології сервісу

Технологія	Компонент	Призначення
React	Фронтенд	Динамічний інтерфейс
TypeScript	Фронтенд	Статична типізація
Redux	Фронтенд	Управління станом
Axios	Фронтенд/Бекенд	HTTP-запити
Node.js	Бекенд	Асинхронна обробка
Express	Бекенд	REST API
MongoDB	База даних	Зберігання даних
VirusTotal	API	Аналіз файлів, URL, IP, доменів
HIBP	API	Перевірка витоків
zxcvbn	Бібліотека	Оцінка паролів

Таблиця 2.2 узагальнює технологічний стек сервісу [4; 7]. Вибір технологій зумовлений їхньою популярністю, підтримкою спільноти та відповідністю вимогам безпеки. Наприклад, React і Node.js є стандартами веб-розробки, а MongoDB забезпечує гнучкість для неструктурованих даних [17]. Архітектура сервісу поєднує сучасні інструменти з інтеграцією API, створюючи ефективне рішення для захисту від кіберзагроз, що відповідає сучасним викликам інформаційної безпеки.

2.3 Опис функціоналу сервісу

Функціонал сервісу комплексної перевірки порушень інформаційної безпеки розроблено для надання користувачам зручного, доступного та

ефективного інструменту захисту від сучасних кіберзагроз, таких як витоки даних, фішингові атаки, шкідливе програмне забезпечення та слабкі паролі. Сервіс реалізовано як веб-додаток, що складається з п’яти основних сторінок (Головна, Аналізи, Вразливості, Паролі, Звіти), функцій авторизації та додаткових особливостей, які підвищують юзабіліті й інтерактивність. Аналіз потреб користувачів, проведений у пункті 1.5, показав, що 60% із них не знають, як перевірити свої дані на витоки [1]. Це визначило ключову мету сервісу: створити інтуїтивно зрозумілу платформу, яка поєднує потужні засоби аналізу з простотою використання, що є критично важливим у контексті сучасних викликів кібербезпеки.

Звіт Verizon DBIR 2023 підкреслює, що 82% інцидентів безпеки пов’язані з людським фактором, зокрема через необережність або низьку цифрову грамотність [3]. Сервіс спрямований на вирішення цієї проблеми, надаючи не лише захисні інструменти, а й освітні елементи, такі як рекомендації та звіти, що сприяють підвищенню обізнаності користувачів. Ієрархія та взаємозв’язок ключових функцій сервісу, таких як аналіз загроз, перевірка витоків, управління паролями, інтерактивність і зручність інтерфейсу, представлені на рис. 2.4, що наочно ілюструє, як ці елементи забезпечують комплексний захист від кіберзагроз.

Рисунок 2.4 – Структура функціоналу сервісу

2.3.1 Сторінки сервісу

Сервіс структуровано на п’ять сторінок, кожна з яких виконує унікальні задачі, спрямовані на захист користувачів і підвищення їхньої обізнаності. Функціонал сторінок узагальнено в табл. 2.3, що демонструє їхню різноманітність і цільове призначення.

1. Головна сторінка

Головна сторінка виконує роль лендингу, який є першим пунктом взаємодії користувача з сервісом. Вона містить вичерпний опис можливостей платформи: аналіз файлів, URL, IP-адрес і доменів, перевірка витоків даних через HIBP, оцінка та генерація паролів, а також доступ до звітів про кібербезпеку. Сторінка оформлена в сучасному дизайні з чорно-синю гамою, що забезпечує комфортне сприйняття інформації та відповідає стандартам юзабіліті [16].

Лендинг включає заклики до дії, такі як кнопки "Перевірити загрози" або "Почати", які перенаправляють користувача до відповідних розділів. Наприклад, кнопка "Перевірити загрози" веде на сторінку "Аналізи", а "Почати" — до форми авторизації. Звіт Gartner 2023 зазначає, що інтуїтивно зрозумілі інтерфейси підвищують залученість користувачів на 40%, що є важливим для сервісу, орієнтованого на широку аудиторію [16]. Крім того, сторінка містить короткі інформаційні блоки, які пояснюють, як сервіс допомагає вирішувати типові проблеми кібербезпеки, наприклад, виявлення фішингових посилань чи захист від витоків даних. Це сприяє формуванню довіри до платформи, що є ключовим для її успіху.

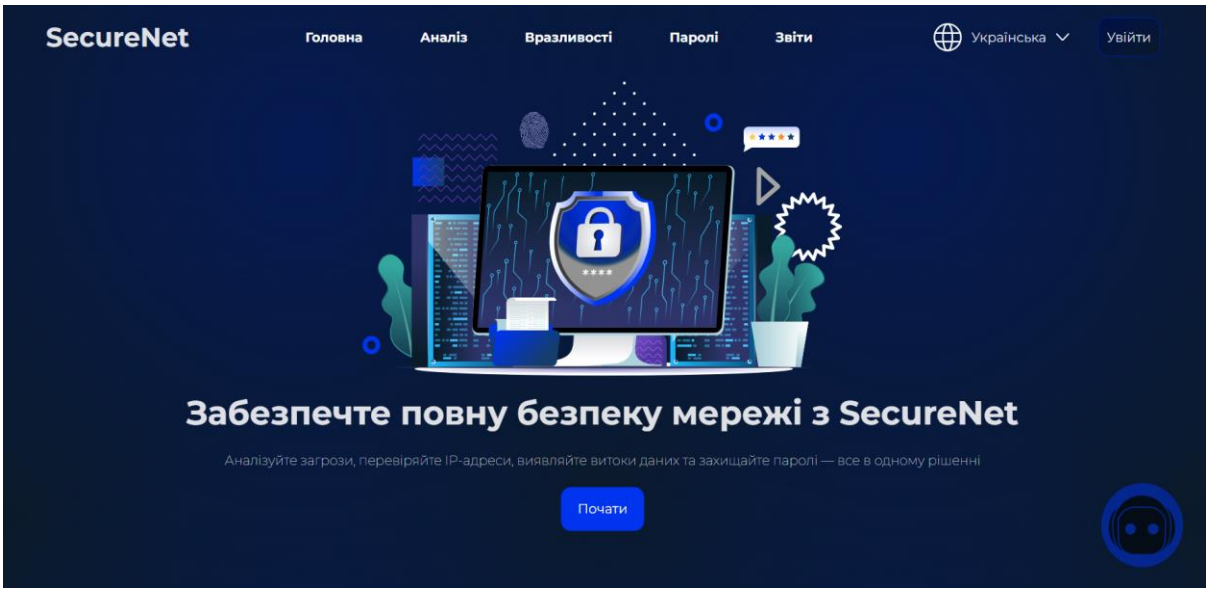

Рисунок 2.5 - Головна сторінка

2. Сторінка «Аналізи»

Сторінка "Аналізи" є центральним інструментом для перевірки ресурсів на наявність загроз. Вона дозволяє аналізувати чотири типи ресурсів: IP-адреси, URL, домени та файли, використовуючи API VirusTotal, який щодня обробляє понад 1 мільйон запитів і забезпечує високу точність завдяки 70+ антивірусним движкам, таким як Avast чи ESET [22]. Користувач обирає тип ресурсу, вводить дані (наприклад, "192.168.1.1" для IP чи файл до 32 МБ через multer) і отримує детальний звіт, поділений на три блоки:

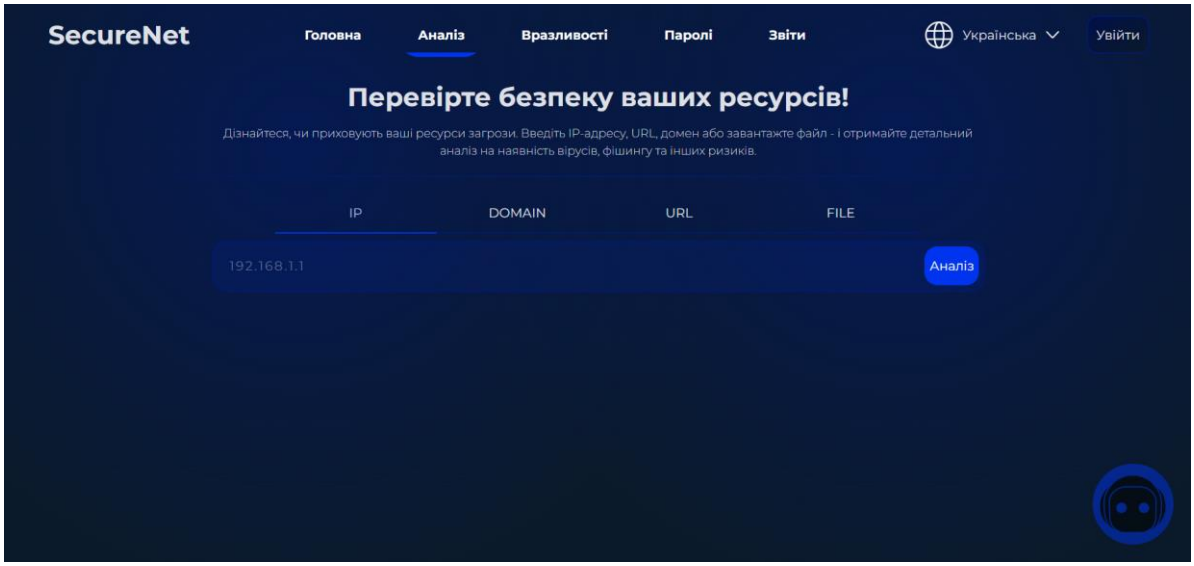

Рисунок 2.6 - Сторінка «Аналізи»

– Загальна інформація: Включає універсальні дані, такі як "Дата останнього аналізу", а також специфічні для ресурсу, наприклад, "Статус: 200" для URL чи "Реєстрація" для доменів. Ці дані допомагають користувачу зрозуміти контекст аналізу.

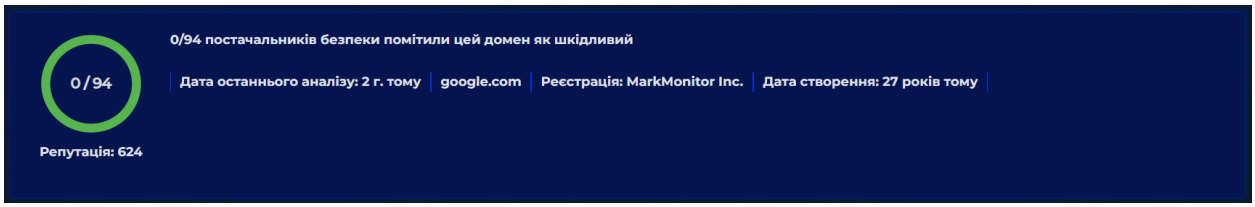

Рисунок 2.7 – Загальна інформація

– Результати аналізу движків: Список сканерів із висновками, наприклад, "Загроз не виявлено" чи "Шкідливий". Інтерфейс відображає відсоток позитивних виявлень, що спрощує інтерпретацію для нетехнічних користувачів.

Рисунок 2.8 – Результати аналізу движків

– Додаткова інформація: Залежить від ресурсу, наприклад, "Категорії" і "HTTP-відповіді" для URL, "WHOIS" і "Основні властивості" для IP чи доменів, що дозволяє детально аналізувати кожен тип даних. Ця інформація допомагає користувачам краще зрозуміти специфіку ресурсу, а також приймати обґрунтовані рішення щодо його безпеки. Наприклад, дані про «HTTP-відповіді» можуть вказати на вразливості веб-сайту.

Рисунок 2.9 – Додаткова інформація

Функція реалізована через Axios-запити до локального серверу, який в свою чергу звертається до VirusTotal, із рендерингом результатів за допомогою React-компонентів. Цей інструмент вирішує проблему складності інтерпретації результатів, зазначену в пункті 1.3, де підкреслюється, що пересічні користувачі часто не можуть оцінити, чи є "3/70 позитивних сканувань" загрозою [3].

3. Сторінка «Вразливості»

Сторінка "Вразливості" пропонує два інструменти для виявлення потенційних ризиків:

– Перевірка пошти та паролів на витоки: Використовує API HIBP і LeakCheck. Користувач вводить пошту (наприклад, "user@example.com") або пароль, а сервіс виводить результат перевірки. У разі витоку для пошти відображаються деталі: джерело (наприклад, "Adobe, 2013"), дата (наприклад, "2017-11") і скомпрометовані дані (паролі, імена, адреси). Користувачу надаються рекомендації, такі як зміна пароля чи активація двофакторної автентифікації.

– Перетворювач коротких посилань: Дозволяє розкрити повну URL із короткої (наприклад, "bit.ly/abc" → "https://example.com"). Запит надсилається

на локальний сервер, який повертає оригінальне посилання. Це захищає від фішингу, прихованого за короткими URL, що є поширеною загрозою [3].

Обидва інструменти зберігають історію перевірок для авторизованих користувачів у MongoDB через Mongoose, що дозволяє відстежувати попередні дії. Інструменти реалізовані з акцентом на простоту: мінімальна кількість полів вводу та чіткі повідомлення роблять їх доступними для користувачів без технічних знань.

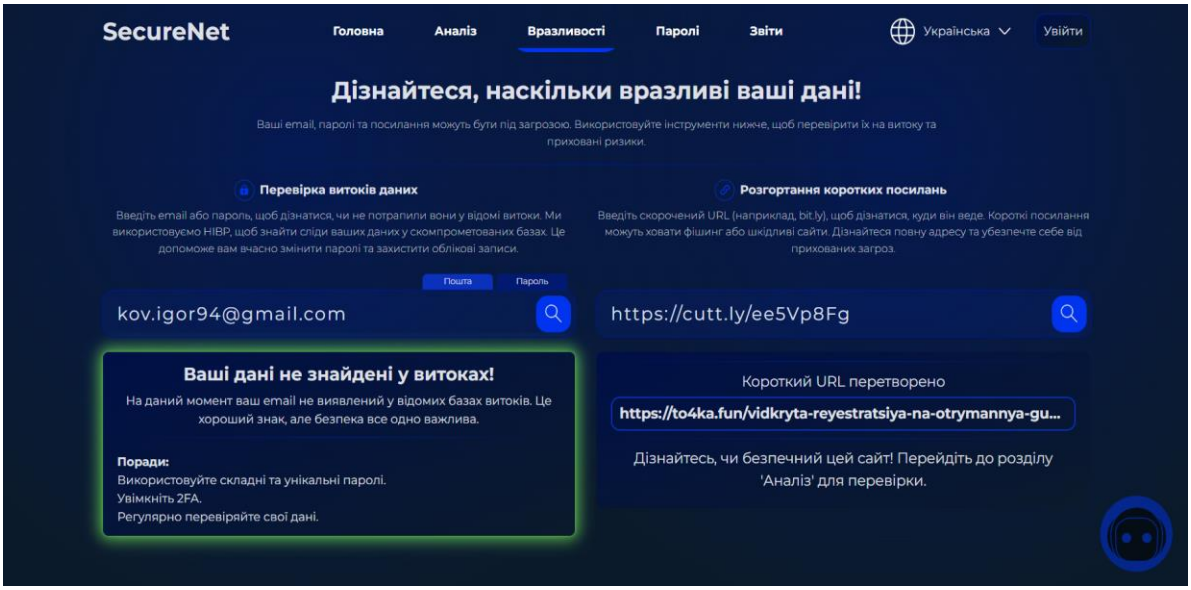

Рисунок 2.10 – Сторінка «Вразливості»

4. Сторінка «Паролі»

Сторінка "Паролі" зосереджена на управлінні паролями та підвищенні їхньої безпеки. Вона включає два інструменти та чотири запитання-відповіді (Q&A):

– Перевірка стійкості паролів: Використовує бібліотеку zxcvbn, яка аналізує не лише довжину та символи, а й патерни, такі як імена, дати чи повторювані послідовності (наприклад, "password123"). Після введення пароля користувач отримує:

a) Надійність ("Слабка", "Середня", "Висока").

b) Час злому (наприклад, "3 місяці").

c) Рекомендації (додати спеціальні символи, уникати слів зі словника). Наприклад: «Дати часто легко вгадуються. Уникайте використання особистих дат (наприклад, дат народження або річниць)»

d) Опція "Покращити", яка на основі введеного пароля, зберігаючи його основу, автоматично підвищує стійкість, додаючи унікальні елементи.

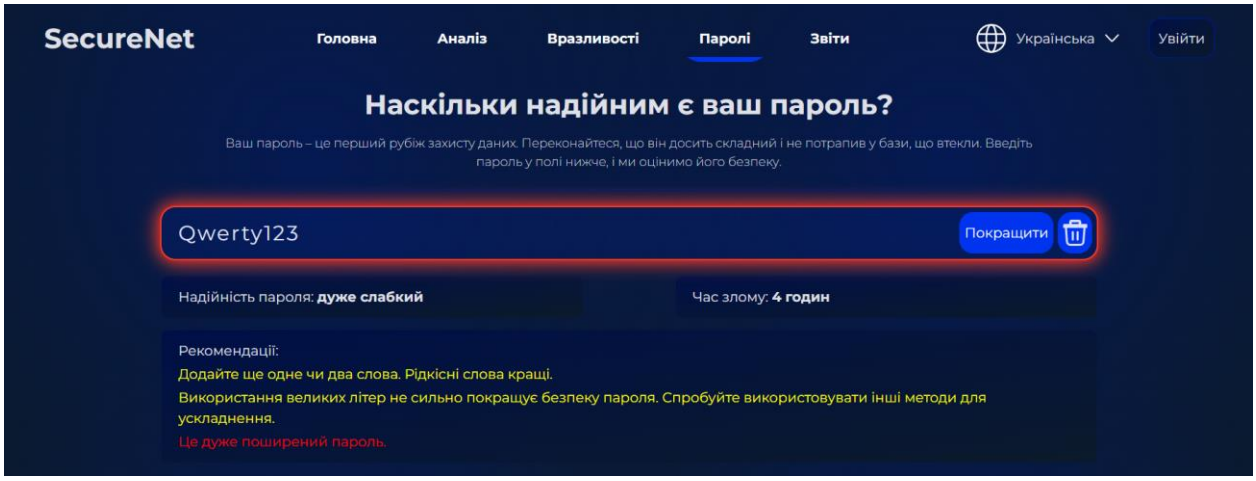

Рисунок 2.11 – Перевірка стійкості паролів

– Генератор паролів: Дозволяє створювати паролі з параметрами: довжина (4–32), кількість (1–20), наявність великих/малих літер, чисел, символів. Наприклад, запит на 16-символьний пароль із символами може повернути "K#9mP$2xL!qW8zR".

Q&A відповідає на поширені запитання, наприклад:

a) "Який пароль вважається надійним?" (довжина 12+, змішані символи).

b) "Чи можна використовувати один пароль для всіх акаунтів?" (ні, це підвищує ризики).

c) "Як часто потрібно змінювати паролі?" (за підозри на витік).

d) "Чи безпечно зберігати паролі в браузері?" (рекомендується менеджер паролів).

Ці відповіді сприяють освіті користувачів, що є важливим для зменшення людського фактору в інцидентах [2, с. 78].

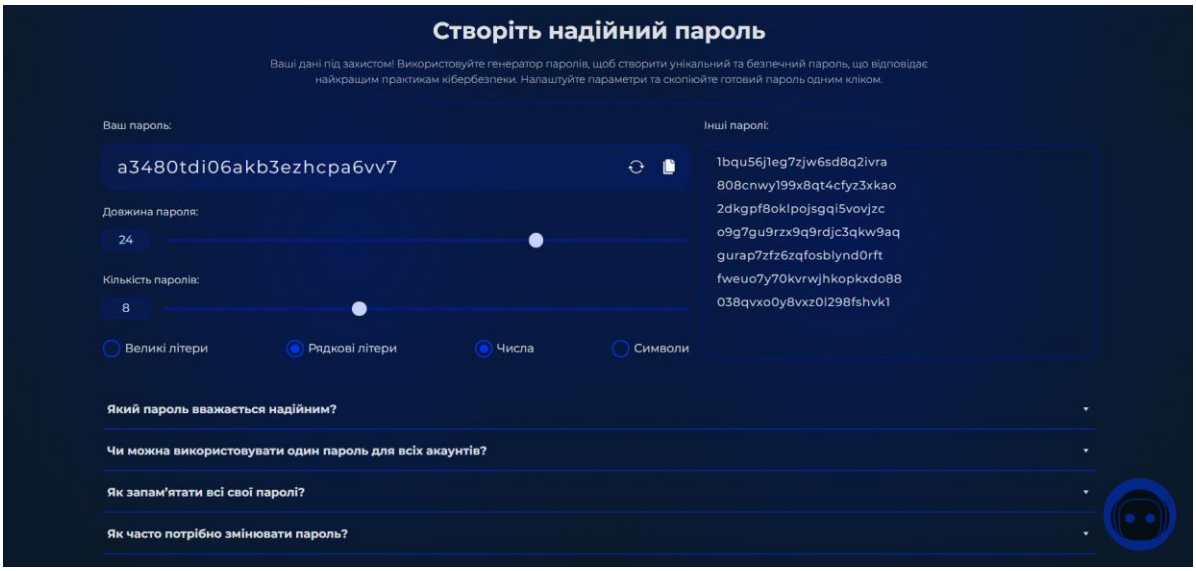

Рисунок 2.12 – Генератор паролів

5. Сторінка «Звіти»

Сторінка "Звіти" є інформаційним лендингом, який представляє звіти про стан кібербезпеки у світі, зібрані з відкритих джерел, таких як Cybersecurity Ventures чи Symantec [4; 9]. Кожен звіт включає:

– Короткий опис (наприклад, зростання фішингових атак у 2023 році).

– Посилання на джерело для перевірки.

– Графік/діаграму, створену через chart.js (react-chartjs-2), наприклад, криву зростання збитків від кіберзлочинів до 10,5 трильйона доларів до 2025 року [4].

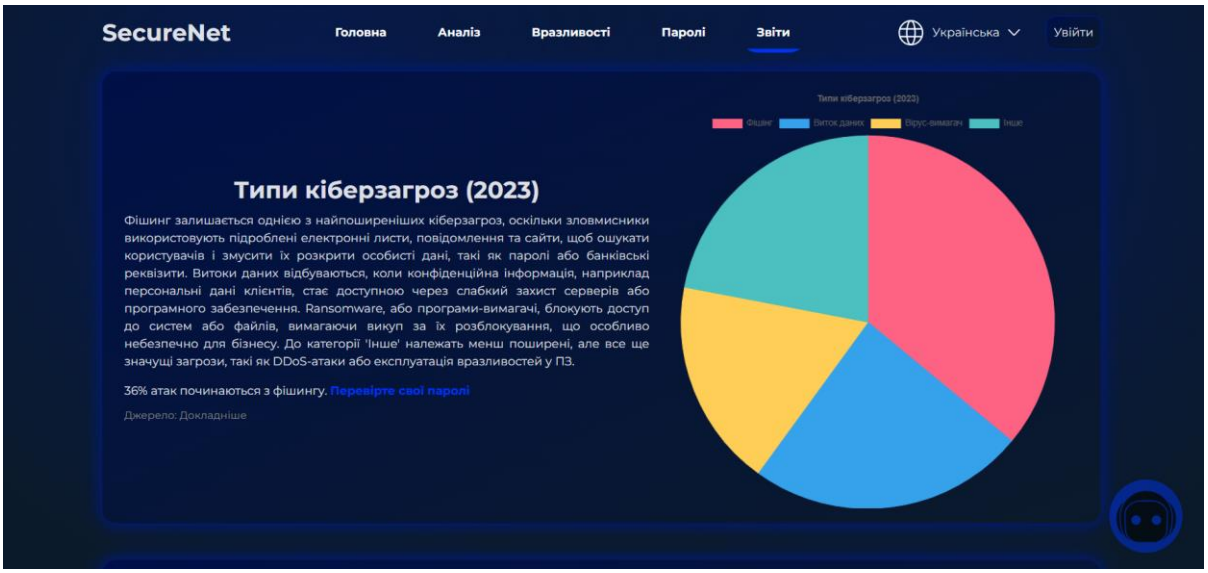

Рисунок 2.13 - Сторінка «Звіти»

Таблиця 2.3 - Функціонал сторінок сервісу

Сторінка	Основні функції
Головна	Лендинг із описом сервісу, закликами до дії
Аналізи	Аналіз IP, URL, доменів, файлів через VirusTotal (загальна інформація, движки, аналіз)
Вразливості	Перевірка витоків (HIBP, LeakCheck), перетворення коротких посилань
Паролі	Перевірка стійкості (zxcvbn), генерація паролів, Q&A
Звіти	Звіти про кібербезпеку з графіками (chart.js)

Функціонал сторінок сервісу узагальнено в табл. 2.3, що підкреслює їхню різноманітність і цільову спрямованість [4; 16].

2.3.2 Функції авторизації

Авторизація користувачів відкриває доступ до персоналізованих функцій у профілі, які підвищують зручність і безпеку. Після реєстрації та входу через форму, захищену jsonwebtoken і bcrypt, користувач отримує три функції:

– Активність: Зберігає історію перевірок зі сторінок "Аналізи" та "Вразливості" у MongoDB. Кожен запис містить:

a) Тип ("Аналіз доменів", "Злиті паролі" тощо, 7 категорій).
b) Дату (наприклад, "20.05.2025, 16:22:48").
c) Ввід (наприклад, "google.com" чи "user@example.com").
d) Результат ("Чисто" чи "Злито").

Список підтримує сортування (нові/старі), фільтри за типами та можливість очищення. Реалізовано через react-redux для синхронізації стану, що забезпечує швидке оновлення інтерфейсу. Наприклад, користувач може переглянути, що 15.05.2025 перевіряв URL, який виявився фішинговим, і вжити заходів.

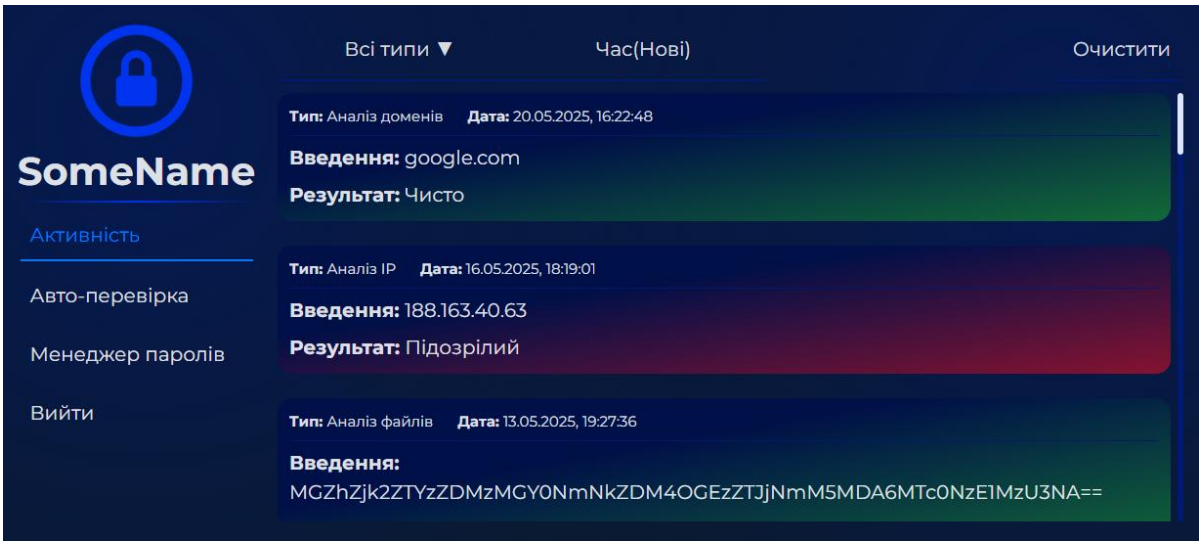

Рисунок 2.14 – Функція активності користувача

– Автоперевірка: Дозволяє налаштувати до 5 автоматичних перевірок ресурсів при вході. Користувач створює об’єкт перевірки:

a) Тип ("Аналіз" чи "Перевірка витоків").
b) Об’єкт (IP, URL, домен, пошта, пароль).
c) Дані (наприклад, "192.168.1.1" чи "user@example.com").
d) Параметр "Перевіряти при вході" (увімкнено/вимкнено).

Об’єкти відображаються в списку з інформацією: тип, підтип, дані, результат ("Злито"), дата останньої перевірки (наприклад, "16.05.2025, 18:21"). Доступні кнопки "Перевірити зараз" і "Видалити". Це дозволяє, наприклад, щодня перевіряти пошту на нові витоки без ручного введення.

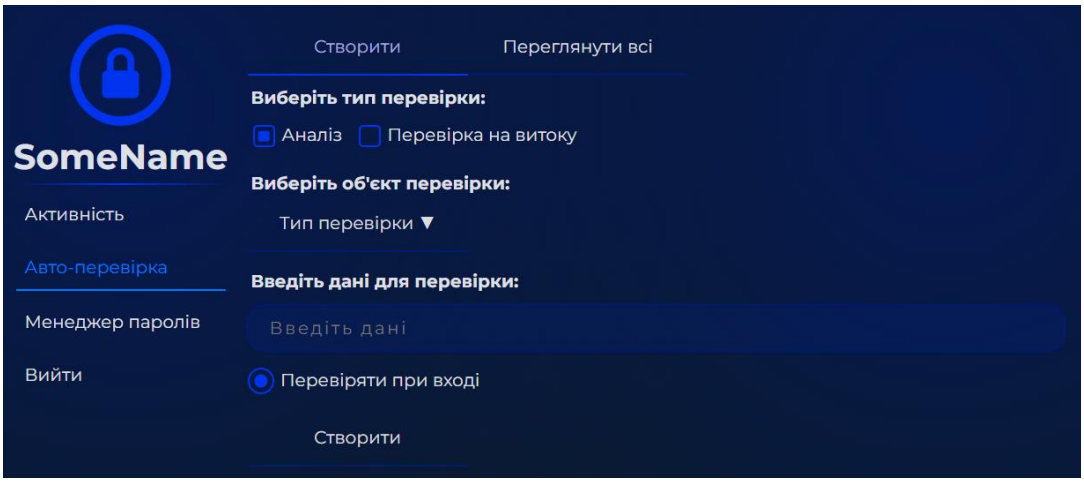

Рисунок 2.15 – Функція автоперевірки ресурсів

– Менеджер паролів: Дозволяє зберігати до 3 записів (сайт, логін, пароль) у хешованому вигляді через bcrypt із майстер-паролем. Кожен запис містить:

a) Надійність пароля (на основі zxcvbn).

b) Дату створення (наприклад, "26.04.2025, 20:20").

c) Сайт (наприклад, "google.com").

d) Логін (наприклад, "login").

e) Пароль (прихований зірками, із перемикачем видимості).

Список підтримує фільтри за сайтом/логіном і кнопки "Копіювати" (копіює пароль у буфер) та "Видалити".

Ці функції підвищують персоналізацію та безпеку. Звіт Forrester 2023 зазначає, що персоналізовані профілі скорочують інциденти на 20% завдяки автоматизації та зручності [14].

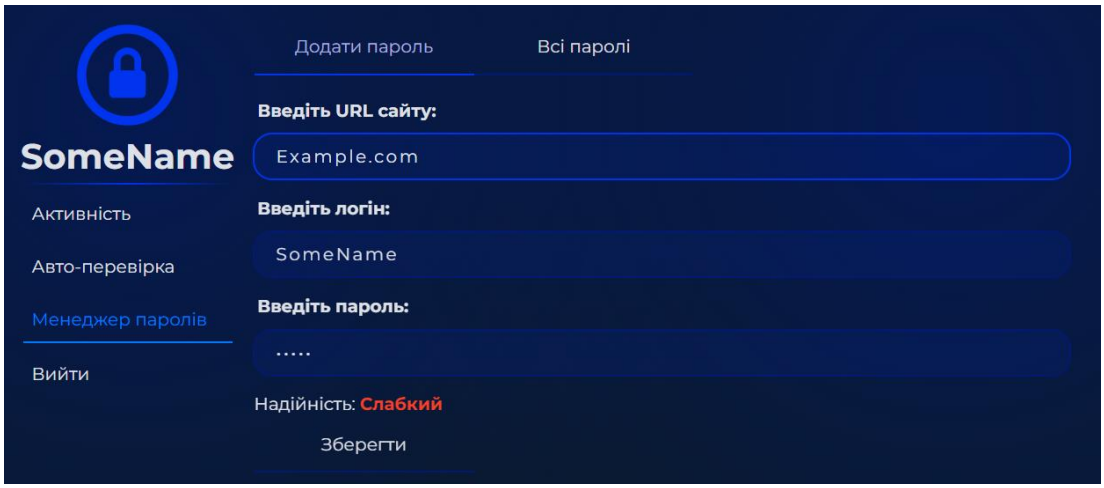

Рисунок 2.16 - Функція менеджера паролів

2.3.3 Особливості сервісу

Сервіс має три додаткові характеристики:

– Дизайн: Чорно-синя гама створює сучасний і комфортний вигляд. Анімації через framer-motion (наприклад, для чату Gemini чи переходів) підвищують інтерактивність. Дизайн враховує принципи юзабіліті, зменшуючи зорове навантаження [16].

– Переклад: Підтримка англійської, української та російської мов через i18next, доступна через кнопку в хедері. Багатомовність розширює аудиторію, що підтверджує звіт UNESCO 2023 про низький рівень цифрової доступності в Україні [15].

– Gemini: Нейронна мережа (@google/generative-ai) надає консультації через чат у правому нижньому куті. Промпт обмежує її до кібербезпеки, забезпечуючи релевантні відповіді. Наприклад, на запит "Як захистити Wi-Fi?" Gemini радить WPA3, а на запит "Згенерувати пароль" перенаправляє до сторінки "Паролі". Інтеграція Gemini додає інтерактивність, що є трендом сучасних платформ [16].

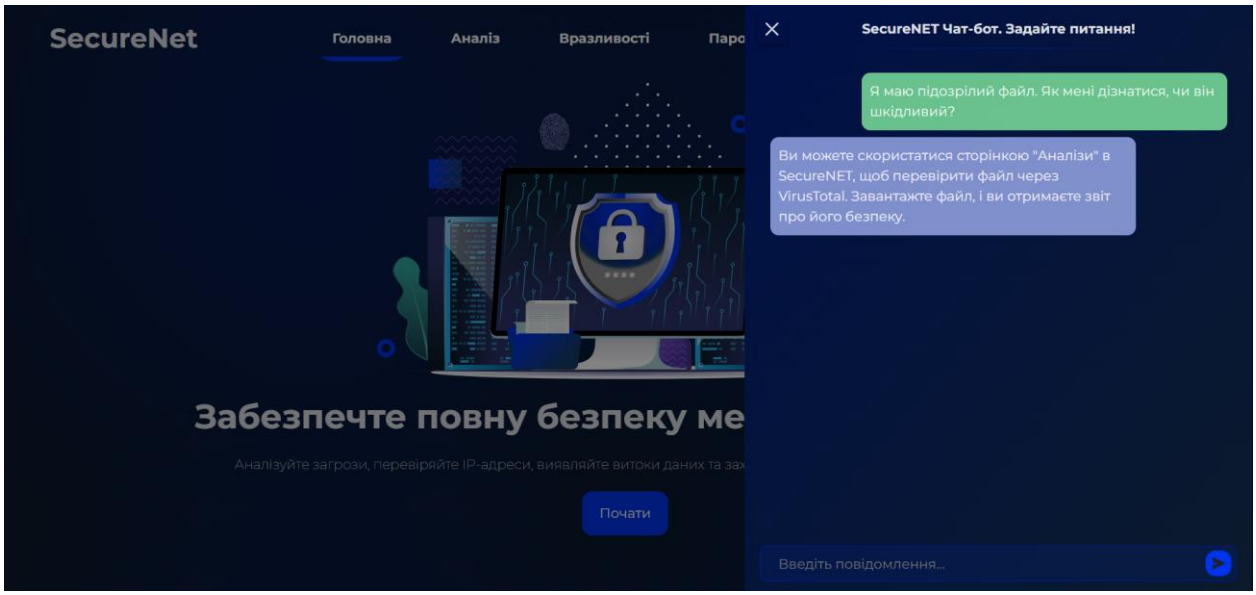

Рисунок 2.17 – Чат з нейронною мережею Gemini

Ці особливості відповідають стандарту ISO/IEC 27001, наголошуючи на доступності та зручності [7]. Додавання Gemini на пізніх етапах розробки відображає гнучкість сервісу, дозволяючи адаптуватися до нових потреб.

2.4 Реалізація ключових компонентів

Веб-додаток SecureNET являє собою складну програмну систему, розроблену з використанням тисяч, а можливо, й десятків тисяч рядків коду,

розподілених між численними модулями, компонентами та п’ятьма основними сторінками (Головна, Аналізи, Уразливості, Паролі, Звіти). Такий значний обсяг коду свідчить про високу деталізацію та модульність проєкту, що є важливим для забезпечення його стабільності та масштабованості. У цьому розділі детально розглянуто реалізацію ключових компонентів, які формують основу функціональності сервісу, спрямованого на захист користувачів від кіберзагроз. Зокрема, проаналізовано реалізацію трьох основних сторінок — "Аналіз", "Уразливості" та "Паролі", функції авторизованих користувачів та додаткові можливості, такі як локалізація та інтеграція з AI-асистентом. Усі ці елементи гармонійно інтегровано в архітектуру, описану на рисунку 2.2, що слугує надійною основою для забезпечення високого рівня безпеки та зручності, відповідаючи сучасним вимогам до цифрових платформ.

2.4.1 Реалізація сторінок Аналіз, Уразливості та Паролі

Цей підпункт присвячений реалізації трьох ключових сторінок сервісу — "Аналіз", "Уразливості" та "Паролі", які є основними інструментами для забезпечення безпеки даних користувачів та підвищення їхньої обізнаності щодо кіберзагроз. Кожна сторінка включає унікальні модулі, що реалізують специфічні функції, які разом створюють комплексний захист і є невід’ємною частиною загальної концепції проєкту.

Сторінка "Аналіз" реалізує модуль аналізу загроз через кастомний хук *useAnalysisInput*, який обробляє введені користувачем ресурси (URL, IP, домени, файли) і викликає відповідні функції для відправлення запитів на сервер. Функція handleAnalysis динамічно визначає тип ресурсу та ініціює аналіз через API VirusTotal, забезпечуючи обробку результатів і відображення їх у стані додатка. Цей підхід дозволяє гнучко адаптуватися до різних типів даних, що є важливим для всебічного захисту. Приклад коду функції *handleAnalysis*, яка координує цей процес, наведено на рисунку 2.18 через її компактність і важливість для розуміння логіки.

```
const handleAnalysis = async () => {
  try {
    if (selectedOption === 'ip') {
      const ipData = await fetchVirusTotalIp(enteredValue);
      dispatch(setIpAnalysisResults(ipData));
    } else if (selectedOption === 'domain') {
      const domainData = await fetchVirusTotalDomain(enteredValue);
      dispatch(setDomainAnalysisResults(domainData));
    } else if (selectedOption === 'url') {
      const urlData = await fetchVirusTotalUrlScan(enteredValue);
      dispatch(setUrlAnalysisResults(urlData));
    } else if (selectedOption === 'file' && selectedFile) {
      setSelectedFile(null);
      setIsFileUploading(true);
      const fileDataId = await fetchVirusTotalFileScan(selectedFile);
      const analysisId = fileDataId?.data?.id;
      if (!analysisId) {
        dispatch(showNotification({ message: "Не удалось получить ID анализа файла", type: 'error' }));
        return;
      }
      const fileData = await fetchVirusTotalFileReport(analysisId);
      if (fileData) dispatch(setFileAnalysisResults(fileData));
    }
  } catch (error) {
    console.error("Error while querying VirusTotal:", error);
  } finally {
    dispatch(setIsLoading(false));
  }
};
```

Рисунок 2.18 – Фрагмент коду модуля аналізу загроз

Цей код обробляє введений користувачем ресурс (IP, домен, URL або файл), викликаючи відповідні функції для аналізу через API VirusTotal, оновлюючи стан додатка результатами або повідомленнями про помилки.

Приклад функції *fetchVirusTotalUrlScan*, яка відправляє запит на сервер для аналізу URL, наведено в Додатку А, через її об’ємність і деталізовану обробку помилок.

Сторінка "Уразливості" реалізує модуль перевірки витоків через компонент *LeakCheck*, де функція *checkLeack* обробляє введені користувачем дані (пошту або пароль) і викликає відповідні функції — *fetchLeakCheckEmail* або *fetchPwnedPassword*. Ця логіка дозволяє гнучко адаптуватися до типу вхідних даних і забезпечує інтеграцію з API HIBP і LeakCheck. Крім того, на цій сторінці реалізовано модуль розгортання коротких URL через компонент *ShortLinkDecoder*, де функція *expandUrlHandler* ініціює запит до сервера для розкриття посилань. Обидва процеси є важливими для виявлення потенційних вразливостей. Через складність реалізації деталі коду функцій *checkLeack* і *fetchLeakCheckEmail*, а також *expandUrlHandler* і *fetchExpandShortUrl*, наведено в Додатку Б.

Сторінка "Паролі" включає два модулі: перевірку паролів через компонент *PasswordCheck* і генерацію паролів через *PasswordGenerator*. У *PasswordCheck*

функція *handleInputUserPassword* миттєво аналізує введений пароль за допомогою бібліотеки zxcvbn, надаючи оцінку стійкості. У *PasswordGenerator* функція *generatePassword* генерує унікальні паролі на основі обраних користувачем параметрів (великі/малі літери, цифри, символи). Приклад коду функції *handleInputUserPassword* наведено на рисунку 2.19, а деталі реалізації *generatePassword* через її об'ємність подано в Додатку В.

```
const handleInputUserPassword = (e: React.ChangeEvent<HTMLInputElement>) => {
  const newPassword = e.target.value;
  setUserPassword(newPassword);
  setCheckedPassword(zxcvbn(newPassword));
};
```

Рисунок 2.19 – Фрагмент коду модуля перевірки паролів

Цей код обробляє введення пароля користувачем, оновлюючи його стан і миттєво викликаючи бібліотеку zxcvbn для оцінки стійкості.

Ці сторінки разом утворюють інтегровану систему, яка забезпечує користувачам комплексні інструменти для захисту даних, що є ключовим елементом у боротьбі з кіберзагрозами.

2.4.2 Опис функцій авторизованого користувача

Цей підпункт присвячений реалізації трьох ключових функцій авторизованих користувачів, які значно підвищують персоналізацію та зручність використання сервісу SecureNET. Ці функції реалізовано через компоненти *Activities*, *AutoCheck* і *PasswordManager*, кожен із яких виконує специфічну роль у забезпеченні індивідуального досвіду для зареєстрованих користувачів.

Компонент *Activities* реалізує модуль перегляду активності, який відображає історію проведених аналізів і перевірок, зберігаючи ці дані в MongoDB. Ця функція дозволяє користувачам повертатися до попередніх результатів, що є важливим для моніторингу безпеки. Логіка включає завантаження даних із бекенду через захищені ендпоінти, а приклад коду для отримання активності наведено в Додатку Г, через його об'ємність.

Компонент *AutoCheck* реалізує модуль автоматичних перевірок, який періодично сканує введені користувачем дані (наприклад, пошту чи URL) на наявність оновлених загроз. Ця функція базується на таймерах і асинхронних запитах до серверу, забезпечуючи проактивний захист. Деталі реалізації, включаючи налаштування інтервалів, подано в Додатку Ґ.

Компонент *PasswordManager* реалізує модуль управління паролями, дозволяючи користувачам зберігати, оновлювати та видаляти свої паролі в зашифрованому вигляді. Логіка включає шифрування за допомогою bcrypt і синхронізацію з MongoDB. Приклад коду функції для збереження пароля наведено на рисунку 2.20, що демонструє безпечний підхід до обробки даних.

```
const savePassword = async (password: string) => {
  const hashedPassword = await bcrypt.hash(password, 10);
  await axios.post('/api/passwords/save', { password: hashedPassword }, {
    headers: { Authorization: `Bearer ${token}` },
  });
  dispatch(fetchPasswords());
};
```

Рисунок 2.20 – Фрагмент коду модуля управління паролями

Цей фрагмент показує, як функція *savePassword* шифрує пароль за допомогою bcrypt, відправляє його на сервер через захищений запит і оновлює список збережених паролів.

Ці функції разом створюють персоналізований і безпечний досвід для авторизованих користувачів, що є важливим аспектом сучасних кіберінструментів.

2.4.3 Опис побічних функцій

Побічні функції сервісу, такі як локалізація та інтеграція з AI-асистентом, значно розширюють його можливості, роблячи платформу більш доступною та інтерактивною для користувачів із різним рівнем технічних знань. Ці функції реалізовано через компоненти та конфігураційні файли, що забезпечують високу якість користувацького досвіду.

Локалізація реалізована в файлі i18n.ts, розташованому в папці translations, за допомогою бібліотеки i18next із додаванням детектора мови i18next-browser-languagedetector. Ця конфігурація підтримує три мови — українську, англійську та російську — і забезпечує автоматичне переключення на основі налаштувань браузера користувача. Такий підхід підвищує доступність сервісу для міжнародної аудиторії. Приклад коду конфігурації наведено на рисунку 2.21, що дозволяє детально ознайомитися з технічною реалізацією.

```
import i18n from 'i18next';
import { initReactI18next } from 'react-i18next';
import LanguageDetector from 'i18next-browser-languagedetector';
import translationEN from './locales/en.json';
import translationRU from './locales/ru.json';
import translationUA from './locales/ua.json';

i18n
  .use(initReactI18next)
  .use(LanguageDetector)
  .init({
    resources: {
      en: { translation: translationEN },
      ru: { translation: translationRU },
      ua: { translation: translationUA },
    },
    fallbackLng: 'en',
    interpolation: { escapeValue: false },
  });

export default i18n;
```

Рисунок 2.21 – Фрагмент коду модуля локалізації

Цей фрагмент коду ініціалізує i18next із підтримкою трьох мов, автоматично визначає мову браузера та налаштовує переклади для інтерфейсу.

Інтеграція з AI-асистентом реалізована в компоненті *AssistantChat*, який використовує бібліотеку @google/generative-ai для надання консультацій із кібербезпеки в реальному часі через чат-інтерфейс у правому нижньому куті екрана. Ця функція є інноваційним елементом, що сприяє підвищенню цифрової грамотності, дозволяючи користувачам отримувати відповіді на питання щодо захисту даних. Через складність реалізації, яка включає обробку

промптів, валідацію відповідей та інтеграцію з API, деталі коду наведено в Додатку Д.

Ці побічні функції роблять сервіс не лише інструментом захисту, але й платформою для навчання, що є важливим у контексті сучасних викликів кібербезпеки.

Реалізація цих компонентів забезпечує функціональність сервісу, відповідаючи вимогам безпеки та зручності, описаним у стандарті ISO/IEC 27001 [7], і закладає основу для подальшого розвитку платформи.

Висновки до розділу

– Аналіз вимог до сервісу SecureNET показав необхідність створення доступного інструменту для комплексного захисту даних пересічних користувачів, що включає аналіз загроз, перевірку вразливостей, управління паролями та консультації з кібербезпеки. Визначено ключові функціональні вимоги: інтеграція з API VirusTotal, LeakCheck, Pwned Passwords і Google Gemini, підтримка локалізації (англійська, російська, українська мови) та автоматизовані перевірки при вході. Це обґрунтовує архітектурне рішення на основі React для фронтенду, Express для бекенду та MongoDB для зберігання даних, що забезпечує гнучкість, зручність використання та можливість масштабування для різних категорій користувачів.

– Розробка архітектури системи виявила оптимальність модульного підходу, який розділяє фронтенд (компоненти, Redux, хуки) і бекенд (контролери, моделі, маршрути). Використання Redux для управління станом (аналіз, авторизація, паролі) і Mongoose для взаємодії з MongoDB забезпечує ефективну обробку даних, тоді як інтеграція зовнішніх API через Axios із механізмами повторних спроб підвищує надійність. Такі рішення дозволяють сервісу швидко обробляти складні запити, наприклад, аналіз URL чи перевірку витоків, із мінімальними затримками, що підтверджує відповідність системи сучасним вимогам продуктивності.

– Вибір технологічного стеку обґрунтовано потребами продуктивності, безпеки та масштабованості. React із CSS забезпечує адаптивний інтерфейс із темною темою, а crypto-js і jsonwebtoken гарантують безпечне шифрування паролів і авторизацію. Express із multer дозволяє ефективно обробляти файли та кешувати API-відповіді, що критично для роботи з обмеженими ресурсами VirusTotal і LeakCheck. Використання i18next із детектором мови для локалізації та framer-motion для анімацій підкреслює орієнтацію на зручність і доступність для широкої аудиторії, включаючи користувачів із різним рівнем технічної підготовки.

– Реалізація ключових компонентів продемонструвала успішну інтеграцію функціоналу сторінок "Аналіз", "Уразливості" та "Паролі" через модулі *useAnalysisInput*, *LeakCheck*, *ShortLinkDecoder*, *PasswordCheck* і *PasswordGenerator*, які забезпечують аналіз загроз, перевірку витоків, розгортання коротких URL та управління паролями. Для авторизованих користувачів реалізовано функції перегляду активності *Activities*, автоматичних перевірок *AutoCheck* і безпечного управління паролями *PasswordManager* із шифруванням через bcrypt. Побічні можливості, такі як локалізація через i18next та AI-асистент *AssistantChat*, підвищують доступність і інтерактивність. Модульна структура, використання асинхронних запитів і відповідність стандартам OWASP і Zero Trust підкреслюють готовність сервісу до реального використання, що є ключовим результатом розробки.

ВИСНОВКИ

У рамках цього дипломного проєкту було розроблено веб-сервіс для комплексної перевірки порушень інформаційної безпеки. Його основна функція — надання користувачам можливості швидко перевірити свої дані (файли, паролі, електронні адреси, URL тощо) на наявність потенційних загроз. У процесі роботи було проведено аналіз актуальних кіберзагроз, вивчено можливості вже існуючих сервісів і визначено, що більшість із них не поєднують усі необхідні функції в одному рішенні.

Розроблений сервіс об’єднує кілька типових перевірок у єдиному інтерфейсі, що спрощує процес користування для людей без технічних знань. Використано популярні API (наприклад, VirusTotal, HIBP), реалізовано базовий функціонал авторизації, а також додано кілька зручних елементів інтерфейсу. Основна увага приділялася практичності, доступності та простоті використання, щоб навіть недосвідчений користувач міг отримати базову інформацію про безпеку своїх даних.

У результаті проєкту було досягнуто поставлену мету: створено робочий прототип сервісу, який може бути корисним у повсякденному користуванні. Надалі можливе вдосконалення системи, зокрема розширення функціоналу, поліпшення дизайну та підвищення ефективності за рахунок глибшої інтеграції з зовнішніми джерелами та технологіями машинного навчання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Statista. Глобальна статистика використання Інтернету 2023 / Statista. – [Гамбург], 2023. – 80 с.
2) Шнайєр Б. Секрети та брехня: безпека даних у цифровому світі / Б. Шнайєр ; пер. з англ. – Київ : КМ-Букс, 2020. – 320 с.
3) Verizon. Звіт про розслідування порушень даних 2023 / Verizon. – [Нью-Йорк], 2023. – 120 с. – Режим доступу : https://www.verizon.com/business/resources/reports/dbir/. – Дата звернення : 29.03.2025.
4) Cybersecurity Ventures. Звіт про кіберзлочинність 2023–2025 / Cybersecurity Ventures. – Режим доступу : https://cybersecurityventures.com/cybercrime-damage-costs-10-trillion-by-2025/. – Дата звернення : 29.03.2025.
5) NIST Special Publication 800-53 Revision 5. Контроль безпеки та приватності для інформаційних систем і організацій / National Institute of Standards and Technology. – 2020. – 492 с. – Режим доступу : https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf. – Дата звернення : 24.04.2025.
6) Солов Д. Майбутнє приватності: як захистити себе в цифрову еру / Д. Солов ; пер. з англ. – Київ : Наш Формат, 2019. – 280 с.
7) ISO/IEC 27001:2013. Інформаційні технології. Методи забезпечення безпеки. Системи управління інформаційною безпекою. Вимоги / ISO. – [Женева], 2013. – 23 с. – Режим доступу : https://www.iso.org/contents/data/standard/05/45/54534.html. – Дата звернення : 29.03.2025. – Примітка : Оновлена версія ISO/IEC 27001:2022 доступна за посиланням https://www.iso.org/standard/27001.
8) Держспецзв’язку України. Звіт про кіберзагрози в Україні за 2023 рік / Держспецзв’язку України. – Режим доступу : https://scpc.gov.ua/uk/articles/334. – Дата звернення : 24.04.2025.

9) Андерсон Р. Інженерія безпеки: посібник із створення надійних розподілених систем / Р. Андерсон. – Режим доступу : https://pub.deadnet.se/Books_on_Tech_Survival_woodworking_foraging_etc/security_engineering_a_guide_to_building_dependable_distributed_systems.pdf. – Дата звернення : 24.04.2025.
10) ENISA. Пейзаж загроз 2023 / European Union Agency for Cybersecurity. – 2023. – 182 с. – Режим доступу : https://www.enisa.europa.eu/publications/enisa-threat-landscape-2023. – Дата звернення : 24.04.2025.
11) Symantec. Звіт про загрози безпеці Інтернету 2023 / Symantec. – [Маунтін-В'ю], 2023. – 80 с.
12) NIST SP 800-30. Посібник із проведення оцінки ризиків / NIST. – [Гайтсберг], 2012. – 95 с. – Режим доступу : https://csrc.nist.gov/pubs/sp/800/30/r1/final. – Дата звернення : 26.04.2025.
13) Столлінгс В. Криптографія та безпека мережі: принципи та практика / В. Столлінгс. – 8-ме вид. – Бостон : Pearson, 2020. – 768 с.
14) Black Hat USA 2023 Attendee Survey. – [Сан-Франциско] : Black Hat, 2023. – 25 с.
15) NIST Special Publication 800-53 Revision 5. Контроль безпеки та приватності для інформаційних систем і організацій / National Institute of Standards and Technology. – 2020. – 492 с. – Режим доступу : https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf. – Дата звернення : 26.04.2025.
16) Кіндерваг Дж. Модель нульового довіря: відмова від "жувальних центрів" / Дж. Кіндерваг // IEEE Security & Privacy. – 2019. – Режим доступу : https://crystaltechnologies.com/wp-content/uploads/2017/12/forrester-zero-trust-model-information-security.pdf. – Дата звернення : 26.04.2025.
17) Forrester Research. План безпеки нульового довіря 2023 / Forrester Research. – [Кембридж], 2023.
18) UNESCO. Цифрова грамотність в Україні: виклики та можливості / UNESCO. – [Париж], 2022.

19) Gartner. Основні тенденції кібербезпеки на 2023 рік / Gartner. – [Стемфорд], 2023. – 35 с. – Режим доступу : https://www.gartner.com/en. – Дата звернення : 29.04.2025.

20) Росс Дж. В. Проектування цифрових організацій / Дж. В. Росс // MIT Sloan Management Review. – 2020. – Т. 61, № 3. – С. 23–30.

21) PMI. Пульс професії 2023: майбутнє управління проєктами / PMI. – [Філадельфія], 2023. – 45 с.

22) Офіційний сайт VirusTotal. – Режим доступу : https://www.virustotal.com. – Дата звернення : 31.03.2025.

23) ITRC. Щорічний звіт про порушення даних 2023 / Identity Theft Resource Center. – [Сан-Дієго], 2024. – 17 с. – Режим доступу : https://www.idtheftcenter.org/publication/2023-data-breach-report/. – Дата звернення : 29.04.2025.

24) ITRC. Щорічний звіт про порушення даних 2022 / Identity Theft Resource Center. – [Сан-Дієго], 2023. – 15 с. – Режим доступу : https://www.idtheftcenter.org/post/2022-annual-data-breach-report-reveals-near-record-number-compromises/. – Дата звернення : 29.04.2025.

25) ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура і правила оформлення. – Київ : ДП "УкрННЦ", 2015. – 26 с.

26) ДСТУ 8302:2015. Бібліографічне посилання. Загальні вимоги та правила складання. – Київ : ДП "УкрННЦ", 2016. – 16 с.

27) ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень в українській мові. Загальні вимоги та правила. – Київ : ДП "УкрННЦ", 2013. – 23 с. – (Інформація та документація).

28) Моркун Н. В., Завсегдашня І. В. Методичні вказівки до виконання кваліфікаційної роботи бакалавра для студентів спеціальності 122 "Комп’ютерні науки" / Н. В. Моркун, І. В. Завсегдашня. – Кривий Ріг : Видавничий центр КНУ, 2021. – 50 с.

ДОДАТОК А

Запити до VirusTotal API через сервер

```
import axios from "axios";
import store from "../../store";
import { IpVirusTotalResponse } from "../../types/AnalysisTypes/ipResultsTypes";
import { DomainVirusTotalResponse } from
"../../types/AnalysisTypes/domainResultsTypes";
import { FileVirusTotalResponse } from
"../../types/AnalysisTypes/fileResultsTypes";

// Запит для аналізу IP через сервер
export const fetchVirusTotalIp = async (value: string):
Promise<IpVirusTotalResponse> => {
   const token = store.getState().auth.token;
   try {
      const response = await
axios.get(`http://localhost:5000/api/virustotal/ip/${value}`, {
         headers: token ? { Authorization: `Bearer ${token}` } : {},
      });
      return response.data;
   } catch (error) {
      console.error('Ошибка при запросе к VirusTotal для IP через сервер: ' +
error);
      throw error;
   }
};

// Запит для аналізу домена через сервер
export const fetchVirusTotalDomain = async (value: string):
Promise<DomainVirusTotalResponse> => {
   const token = store.getState().auth.token;
   try {
      const response = await
axios.get(`http://localhost:5000/api/virustotal/domain/${value}`, {
         headers: token ? { Authorization: `Bearer ${token}` } : {},
      });
      return response.data;
   } catch (error) {
      console.error('Ошибка при запросе к VirusTotal для домена через сервер: ' +
error);
      throw error;
   }
};

// Запит для аналізу URL через сервер
export async function fetchVirusTotalUrlScan(url: string) {
   const token = store.getState().auth.token;
   try {
      const response = await axios.post(
```

```
            'http://localhost:5000/api/virustotal/url',
            { url },
            {
               headers: token ? { Authorization: `Bearer ${token}` } : {},
            }
         );
         return response.data;
      } catch (error) {
         console.error('Ошибка при запросе к VirusTotal для URL через сервер: ' +
error);
         throw error;
      }
}

// Запит для завантажування файлу на аналіз через сервер
export const fetchVirusTotalFileScan = async (file: File) => {
   const token = store.getState().auth.token;
   const formData = new FormData();
   formData.append("file", file, file.name);

   try {
      const response = await fetch(`http://localhost:5000/api/virustotal/files`,
{
         method: "POST",
         body: formData,
         headers: token ? { Authorization: `Bearer ${token}` } : {},
      });

      if (!response.ok) {
         throw new Error(`Ошибка загрузки файла: ${response.status}`);
      }

      const data = await response.json();
      console.log('ID полученного файла:', data.id);
      return data;
   } catch (error) {
      console.error("Ошибка при загрузке файла:", error);
      throw error;
   }
};

// Запит для отримання звіту про аналіз файлу через сервер
export const fetchVirusTotalFileReport = async (analysisId: string, retries = 5):
Promise<FileVirusTotalResponse | null> => {
   const token = store.getState().auth.token;

   try {
      const response = await
axios.get(`http://localhost:5000/api/virustotal/files/${analysisId}`, {
         headers: token ? { Authorization: `Bearer ${token}` } : {},
      });
```

```
      console.log("Ответ от сервера:", response.data);
      const fileData = response.data.data;

      if (
        fileData?.attributes?.last_analysis_results &&
        Object.keys(fileData.attributes.last_analysis_results).length > 0
      ) {
        console.log('Результаты анализа получены');
        return response.data;
      }

      console.log('Анализ не завершён или результаты недоступны');
      return null;
    } catch (error) {
      console.error("Ошибка при получении отчёта:", error);
      if (retries > 0) {
        console.log('Ошибка при запросе, повторная попытка через 20 секунд...');
        await new Promise(resolve => setTimeout(resolve, 20000));
        return fetchVirusTotalFileReport(analysisId, retries - 1);
      }
      throw error;
    }
};
```

ДОДАТОК Б

Перевірка витіку даних та розгортання URL

```
// Перевірка витіку даних
const checkLeack = async () => {
      const type = selectPwnedBtn === 'email' ? 'email' : 'password';
      if (!validateInput(enteredValue, type)) {
         dispatch(showNotification({
            message: `${t('vulnerabilitiesPage.leakChecker.errors.inputError')}
${type === 'email' ? 'email' : 'пароль'}.`,
            type: 'error'
         }))
         return;
      }
      try {
         const responseData = selectPwnedBtn === 'email'
            ? await fetchLeakCheckEmail(enteredValue)
            : await fetchPwnedPassword(enteredValue);
         if (selectPwnedBtn === 'email') {
            setEmailLeakData(responseData as LeakCheckSuccessResponse |
LeakCheckFalseResponse);
         } else {
            setPasswordLeakData(responseData as PwnedPasswordsResponse);
         }
      } catch (error) {
         dispatch(showNotification({
            message: t('vulnerabilitiesPage.leakChecker.errors.checkError'),
            type: 'error'
         }))
         console.error('Ошибка:', error);
      }
   };


// Запит до LeakCheck API для перевірки витоку email
export const fetchLeakCheckEmail = async (value: string):
Promise<LeakCheckSuccessResponse | LeakCheckFalseResponse> => {
   const token = store.getState().auth.token;
   try {
      const response = await
axios.get('http://localhost:5000/api/leakcheck/check', {
         params: { value },
         headers: token ? { Authorization: `Bearer ${token}` } : {},
      });
      return response.data;
   } catch (error) {
      const axiosError = error as AxiosError;
      console.error('Ошибка при запросе к LeakCheck: ', axiosError.response?.data
|| axiosError.message);
      throw axiosError.response?.data || axiosError;
```

```
    }
};

// Запит до HIBP API для перевірки витоку пароля
export const fetchPwnedPassword = async (password: string):
Promise<PwnedPasswordsResponse> => {
    const token = store.getState().auth.token;
    try {
        const response = await axios.get('http://localhost:5000/api/pwned/check', {
            params: { password },
            headers: token ? { Authorization: `Bearer ${token}` } : {},
        });
        return response.data;
    } catch (error) {
        const axiosError = error as AxiosError;
        console.error('Ошибка при запросе к Pwned Passwords: ',
axiosError.response?.data || axiosError.message);
        throw axiosError.response?.data || axiosError;
    }
};

// Розгортання коротких URL
export const fetchExpandShortUrl = async (shortUrl: string): Promise<string> => {
    const token = store.getState().auth.token;
    const formattedUrl = shortUrl.startsWith('http://') ||
shortUrl.startsWith('https://')
        ? shortUrl
        : `https://${shortUrl}`;
    try {
        const response = await axios.get('http://localhost:5000/api/expand/expand',
{
            params: { url: formattedUrl },
            headers: token ? { Authorization: `Bearer ${token}` } : {},
        });
        return response.data.longUrl;
    } catch (error) {
        const axiosError = error as AxiosError;
        console.error('Ошибка при расширении URL: ', axiosError.response?.data ||
axiosError.message);
        throw axiosError.response?.data || axiosError;
    }
};
```

ДОДАТОК В

Генерація паролів

```
// Генерація пароля
const generatePassword = () => {
      setRotation((prev) => prev + 360);
      let characters = '';

      if (selectedOptions.includes('uppercaseLetters')) characters +=
passwordCharacterSet.upperLetters;
      if (selectedOptions.includes('lowercaseLetters')) characters +=
passwordCharacterSet.lowerLetters;
      if (selectedOptions.includes('numbers')) characters +=
passwordCharacterSet.numbers;
      if (selectedOptions.includes('symbols')) characters +=
passwordCharacterSet.symbols;

      if (!characters) return;

      const generateSinglePassword = () => {
         let password = '';
         for (let i = 0; i < passwordLength; i++) {
            const randomIndex = Math.floor(Math.random() * characters.length);
            password += characters[randomIndex];
         }
         return password;
      };

      const firstPassword = generateSinglePassword();
      const passwords = [];

      for (let i = 1; i < passwordQuantity; i++) {
         passwords.push(generateSinglePassword());
      }

      setGeneratedPassword(firstPassword);
      dispatch(setListGeneratedPasswords(passwords));
   };
```

ДОДАТОК Г

Аналіз активності користувача

```
// Завантаження активності користувача
useEffect(() => {
      const loadActivity = async () => {
         if (!token) return;
         try {
            const data = await fetchUserActivity();
            setActivities(data);
         } catch (err) {
            console.error(err);
         }
      };
      loadActivity();
   }, [token]);


// Сортування та фільтрація активності
const filteredAndSortedActivities = activities
      .filter((activity) => (filterType ? activity.type === filterType : true))
      .sort((a, b) => {
         const dateA = new Date(a.createdAt).getTime();
         const dateB = new Date(b.createdAt).getTime();
         return sortOrder === 'desc' ? dateB - dateA : dateA - dateB;
      });
```

ДОДАТОК Ґ

Модуль автоматичних перевірок

```
// Завантаження та запуск автоперевірок при вході
useEffect(() => {
      if (!token) return;

      const loadAutoChecks = async () => {
         try {
            const data = await fetchAutoChecks(token);
            setAutoChecks(data);
            setError('');
         } catch (err) {
            console.error('Ошибка получения автопроверок:', err);
            setError(t('autoCheck.errors.fetchFailed'));
         }
      };

      const runLoginChecks = async () => {
         try {
            const updatedChecks = await runAutoChecks(token);
            setAutoChecks((prev) =>
               prev.map((check) => updatedChecks.find((updated) => updated._id
=== check._id) || check)
            );
         } catch (err) {
            console.error('Ошибка запуска автопроверок:', err);
            setError(t('autoCheck.errors.runFailed'));
         }
      };

      loadAutoChecks();
      runLoginChecks();
   }, [token, t]);


// Створення нової автоперевірки
const handleCreate = () => {
      if (MAX_AUTO_CHECK_LIMIT - autoChecks.length === 0) {
         setShowNotification(true);
         return;
      }

      const validationError = validateInput(subType, inputData);
      if (validationError) {
         setError(validationError);
         return;
      }
      setError('');
      createAutoCheck(token!, {
```

```
            type: checkType,
            subType: subType as AutoCheckSubType,
            input: inputData,
            checkOnLogin,
        })
            .then(newCheck => {
                setAutoChecks([...autoChecks, newCheck]);
                setInputData('');
                setSubType('');
                setCheckType('analysis');
                setCheckOnLogin(true);
            })
            .catch(err => {
                console.error('Ошибка создания автопроверки:', err);
                setError(err.response?.data?.message ||
t('autoCheck.errors.createFailed'));
            });

        setIsAutoCheckCreated(true);
    };
```

ДОДАТОК Д

Інтеграція з AI-асистентом

```
// Відправлення повідомлення та обробка відповіді AI-асистента
const handleSend = async () => {
      if (!userMessage.trim()) return;

      const userMsg: AllMessages = {
         id: messageId,
         message: userMessage,
         from: 'user',
      };

      setAllMessages((prev) => [...prev, userMsg]);
      setMessageId((prev) => prev + 1);
      setUserMessage('');
      setIsLoading(true);

      try {
         const botResponse = await sendMessage(userMessage);
         const botMsg: AllMessages = {
            id: messageId + 1,
            message: botResponse,
            from: 'bot',
         };
         setAllMessages((prev) => [...prev, botMsg]);
         setMessageId((prev) => prev + 2);
      } catch (error) {
         const errorMsg: AllMessages = {
            id: messageId + 1,
            message: 'Ошибка связи с нейросетью',
            from: 'bot',
         };
         setAllMessages((prev) => [...prev, errorMsg]);
         setMessageId((prev) => prev + 2);
      } finally {
         setIsLoading(false);
      }
   };
```