Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Розробка системи моніторингу вразливостей з NVD із фільтрацією за Common Platform Enumeration та автоматичними сповіщеннями

Виконав(ла): студент(ка) 6 курсу, групи СПм-61
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Бурса В. В.
(підпис) (прізвище та ініціали)

Керівник _____ Мудрик І. Я.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Стоянов Ю. М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Петрик М. Р.
(підпис) (прізвище та ініціали)

Рецензент _____ Луцик Н. С.
(підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М. Р.
(підпис) (прізвище та ініціали)
« » 20___ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр
(назва освітнього ступеня)

за спеціальністю 121 «Інженерія програмного забезпечення»
(шифр і назва спеціальності)

студенту Бурсі Валерії Віталіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка системи моніторингу вразливостей з NVD із фільтрацією за Common Platform Enumeration та автоматичними сповіщеннями

Керівник роботи Мудрик Іван Ярославович, доктор філософії, доцент кафедри ПІ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*27*» *листопада* 2025_ року № *4/7-1045*

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи відкриті дані про вразливості з NVD (API/JSON feeds) та формати ідентифікації CPE 2.3 і CVSS, а також задані параметри моніторингу (перелік CPE/ключових слів, поріг важливості, інтервал перевірки) і програмне середовище реалізації

4. Зміст роботи (перелік питань, які потрібно розробити)
Аналіз предметної області, існуючих рішень та постановка задачі, формування вимог до системи, проектування архітектури та бази даних системи, реалізація основних модулів системи впровадження та тестування системи, аналіз отриманих результатів та оцінка ефективності

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
діаграма варіантів використання; структура таблиць бази даних SQLite; діаграма класів; головна сторінка налаштувань та списку моніторингів; скріншоти результатів тестування; скріншот структури проекту на GitHub; слайд 1: Титульний; слайд 2: Актуальність тематики; слайд 3: Аналіз існуючих рішень; слайд 4: Мета та задачі проекту; слайд 5: Вимоги системи; слайд 6: Діаграма варіантів використання; слайд 7: Діаграма класів; слайд 8: Ключові модулі системи; слайд 9: Візуалізація результату розробки; слайд 10: Результати тестування системи; слайд 11: Практичні результати та новизна рішення; слайд 12: Апробація результатів та публікації; слайд 13: Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Збір і аналіз наукових, нормативних джерел, уточнення	28.11.2025	**Виконано**
	теми, мети, задач і предмета дослідження		
2	Аналіз предметної області, огляд існуючих рішень	30.11.2025	**Виконано**
	та формування вимог до системи		
3	Проектування системи: архітектура, структура бази	02.12.2025	**Виконано**
	даних, моделювання		
4	Розробка програмного забезпечення	04.12.2025	**Виконано**
5	Налаштування та впровадження: конфігурація	06.12.2025	**Виконано**
	середовища, підключення Gmail, автоматизація запуску		
	перевірок		
6	Тестування системи за ключовими сценаріями, аналіз	08.12.2025	**Виконано**
	результатів і доопрацювання		
7	Оцінка ефективності, формування висновків та	10.12.2025	**Виконано**
	рекомендацій щодо практичного використання і		
	розвитку системи		
8	Підготовка розділу охорони праці та безпеки в	11.12.2025	**Виконано**
	надзвичайних ситуаціях		
9	Оформлення пояснювальної записки, списку джерел та	12.12.2025	**Виконано**
	графічних матеріалів		
10	Захист кваліфікаційної роботи	22.12.2025	

Студент _____ Бурса В. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Мудрик І. Я.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бурса Валерія Віталіївна. Розробка системи моніторингу вразливостей з NVD із фільтрацією за Common Platform Enumeration та автоматичними сповіщеннями. Кваліфікаційна робота освітнього ступеня «магістр». Тернопільський національний технічний університет імені Івана Пулюя, кафедра програмної інженерії, група СПм-61, спеціальність 121 «Інженерія програмного забезпечення». Тернопіль, 2025. Сторінок 76, таблиць 0, рисунків 25, додатків 2, бібліографічних посилань 44.

Метою роботи є створення веборієнтованої системи автоматизованого моніторингу вразливостей програмних продуктів на основі даних National Vulnerability Database з фільтрацією за Common Platform Enumeration та надсиланням e-mail сповіщень.

Об’єктом дослідження є процес моніторингу вразливостей програмного забезпечення за даними публічних баз. Предметом дослідження є методи та програмні засоби побудови такої системи з використанням NVD, CPE та електронної пошти. У роботі застосовано аналіз стандартів і форматів даних, проєктування вебсистем і баз даних, реалізацію серверної частини на Python/Flask з використанням SQLite та функціональне тестування.

У роботі виконано аналіз предметної області та існуючих рішень, сформульовано вимоги до системи, спроєктовано архітектуру, базу даних і основні модулі для отримання даних з NVD, фільтрації вразливостей та формування сповіщень. Описано впровадження та тестування системи, а також оцінено ефективність запропонованого підходу. Окремо розглянуто фактори ризику для користувачів комп’ютерної мережі, що працюють із системою, та наведено основні заходи безпеки.

Ключові слова: кібербезпека, моніторинг вразливостей, National Vulnerability Database, Common Platform Enumeration, CPE, автоматичні сповіщення, Python, Flask, SQLite, електронна пошта.

ABSTRACT

Bursa Valeriia Vitaliivna. Development of a vulnerability monitoring system based on NVD with Common Platform Enumeration filtering and automatic notifications. Qualification work of the educational level “Master”. Ternopil Ivan Puluj National Technical University, Department of Software Engineering, Group SPm-61, Specialty 121 “Software Engineering”. Ternopil, 2025. Pages 76, tables 0, figures 25, annexes 2, bibliographic references 44.

The purpose of the work is to create a web-based system for automated monitoring of software vulnerabilities using data from the National Vulnerability Database with filtering by Common Platform Enumeration and sending e-mail notifications.

The object of the study is the process of monitoring software vulnerabilities based on public security databases. The subject of the study is the methods and software tools for building such a system using NVD, CPE and e-mail. The research applies analysis of standards and data formats, design of web systems and databases, implementation of the server side in Python/Flask with SQLite, and functional testing.

The work presents an analysis of the problem domain and existing solutions, formulates system requirements, and designs the architecture, database and core modules for retrieving data from NVD, filtering vulnerabilities and generating notifications. The implementation and testing of the system are described, and the effectiveness of the proposed approach is evaluated. A separate part considers risk factors for computer network users working with the system and outlines basic safety measures.

Keywords: cyber security, vulnerability monitoring, National Vulnerability Database, Common Platform Enumeration, CPE, automatic notifications, Python, Flask, SQLite, e-mail.

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ

БД	база даних.
БЖД	безпека життєдіяльності.
ІБ	інформаційна безпека.
СКБД	система керування базами даних.
API	Application Programming Interface, програмний інтерфейс для взаємодії між компонентами програмних систем.
CPE	Common Platform Enumeration, стандарт ідентифікації програмних та апаратних платформ.
CVE	Common Vulnerabilities and Exposures, загальноприйнятий реєстр відомих вразливостей.
CVSS	Common Vulnerability Scoring System, система бального оцінювання критичності вразливостей.
GUI	Graphical User Interface, графічний інтерфейс користувача.
JSON	JavaScript Object Notation, текстовий формат обміну структурованими даними.
NIST	National Institute of Standards and Technology, Національний інститут стандартів і технологій США.
NVD	National Vulnerability Database, національна база даних вразливостей, що підтримується NIST.
ORM	Object Relational Mapping, підхід до відображення об’єктів програми в реляційну базу даних.
SMTP	Simple Mail Transfer Protocol, протокол передавання електронної пошти.

ЗМІСТ

ВСТУП

Сучасний стан кібербезпеки характеризується стрімким зростанням кількості оприлюднених вразливостей програмного забезпечення, у тому числі для масово використовуваних платформ, операційних систем та вебзастосунків. Публічні бази даних, такі як National Vulnerability Database, забезпечують відкритий доступ до структурованої інформації про відомі вразливості, однак обсяг цих даних постійно збільшується. В умовах обмежених ресурсів невеликих організацій та окремих фахівців ручний моніторинг таких баз стає малоефективним, потребує значних часових витрат і створює ризик несвоєчасного виявлення критичних загроз. Особливої актуальності набувають програмні засоби, які здатні автоматизувати відбір релевантних записів, зосереджуючись саме на тих продуктах, що реально використовуються в інформаційній інфраструктурі.

Важливу роль у формалізації опису програмних і апаратних платформ відіграє стандарт Common Platform Enumeration, що дозволяє однозначно позначати продукти у вигляді CPE-рядків. Використання CPE дає змогу автоматично співвідносити конкретні елементи інфраструктури з записами про вразливості у базі NVD та інших джерелах. Проте у практичній діяльності фахівці з безпеки часто змушені самостійно налаштовувати фільтри, експорт даних та обробку результатів, що ускладнює оперативний моніторинг. Тому постає задача розробки простої у впровадженні системи, яка поєднує можливості NVD, формальний опис продуктів через CPE та автоматичне сповіщення відповідальних осіб про нові релевантні вразливості.

Метою кваліфікаційної роботи є розробка системи моніторингу вразливостей з бази NVD із фільтрацією за Common Platform Enumeration та автоматичними сповіщеннями електронною поштою заздалегідь налаштованим отримувачам. Для досягнення поставленої мети необхідно вирішити комплекс взаємопов’язаних задач. Потрібно виконати аналіз предметної області та існуючих підходів до моніторингу вразливостей за даними публічних баз. Необхідно сформулювати вимоги до програмної системи, визначити її функціональні можливості та

обмеження. На основі вимог слід розробити архітектуру системи, спроєктувати структуру бази даних і основні програмні компоненти, що забезпечують взаємодію з NVD, фільтрацію результатів і надсилання сповіщень. Важливо реалізувати веборієнтований інтерфейс для налаштування моніторингів та списків отримувачів, а також виконати тестування системи і провести оцінку її ефективності у типових сценаріях використання.

Об’єктом дослідження у роботі є процес моніторингу вразливостей програмних продуктів на основі публічних баз даних безпеки. Предметом дослідження є методи та програмні засоби автоматизованого моніторингу вразливостей з NVD із використанням фільтрації за Common Platform Enumeration та механізмів автоматичного сповіщення відповідальних осіб.

Наукова новизна отриманих результатів полягає у поєднанні формалізованого опису програмних продуктів за допомогою CPE з гнучкою конфігурацією «профілів моніторингу», що дозволяє налаштовувати незалежні сценарії відстеження для різних груп продуктів та команд. У роботі запропоновано підхід до відбору релевантних записів з NVD, який одночасно враховує відповідність CPE-ідентифікаторам та додаткові ключові слова, що описують технології або компоненти. Реалізовано алгоритм формування консолідованих e-mail звітів, орієнтованих не на повний перелік знайдених вразливостей, а на стислу добірку важливих записів з урахуванням заданого порогу критичності. На відміну від існуючих важковагових рішень, акцент зроблено на простоті архітектури та можливості розгортання системи на окремій робочій станції без складної інфраструктури, що розширює потенціал її застосування у невеликих командах, навчальних лабораторіях та індивідуальній практиці фахівця.

Практичне значення роботи полягає у створенні готового програмного засобу, який може бути використаний як допоміжний інструмент у процесі управління вразливостями. Розроблена система дає змогу скоротити час, який витрачається на щоденний перегляд бази NVD, зменшити ризик пропуску критичних вразливостей та систематизувати процес отримання інформації про загрози. Вона може бути інтегрована у робочі процеси невеликої організації, групи

інформаційної безпеки чи навчального підрозділу, а також слугувати базою для подальшого розвитку функціональності, зокрема підключення додаткових джерел даних, альтернативних каналів сповіщення та аналітичних модулів.

Апробація результатів кваліфікаційної роботи здійснювалася шляхом підготовки та представлення тез доповіді на тему «Перспективи автоматизованого оповіщення про вразливості програмних продуктів на основі даних NVD та ідентифікаторів CPE» у межах XIII науково технічної конференції «Інформаційні моделі, системи та технології». У тезах було викладено мотивацію вибору теми, сформульовано основну ідею створення автоматизованого сервісу оповіщення, окреслено підхід до використання CPE та NVD, а також визначено перспективи подальшого розвитку системи. Представлення результатів на науковому заході підтвердило актуальність обраної проблематики та викликало зацікавлення серед учасників конференції.

За матеріалами кваліфікаційної роботи підготовлено тези доповіді, опубліковані у збірнику матеріалів XIII науково технічної конференції «Інформаційні моделі, системи та технології» (див. додаток А). Отримані результати можуть бути використані як основа для подальших наукових публікацій, а також для удосконалення і розширення розробленої системи у напрямку інтеграції з іншими засобами кіберзахисту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

У сучасному цифровому середовищі питання забезпечення безпеки інформаційних систем стає дедалі важливішим. Зростаюча кількість вразливостей у програмному забезпеченні та постійне оновлення відповідних даних вимагають системного підходу до моніторингу, аналізу та своєчасного реагування на нові загрози. Наявність структурованих баз даних вразливостей, таких як NVD, значно полегшує доступ до відповідної інформації, однак самостійна обробка цих даних у ручному режимі є складною і неефективною.

У цьому розділі буде сформульовано основні положення предметної області, розглянуто ключові терміни та стандарти, проведено аналіз існуючих підходів і інструментів моніторингу вразливостей. Також будуть окреслені вимоги до програмної системи, яка мала би вирішувати відповідні завдання, та представлено базову модель сценаріїв її використання.

1.1 Постановка проблеми та предметної області

Розвиток цифрових технологій та стрімке зростання обсягів програмного забезпечення, яке використовується в бізнесі, державних установах і повсякденному житті, призвели до зростання залежності від інформаційних систем. З іншого боку, збільшення кількості компонентів у програмній інфраструктурі одночасно підвищує ризик появи вразливостей – програмних помилок або конструктивних слабких місць, що можуть бути використані зловмисниками для несанкціонованого доступу, порушення роботи систем чи компрометації даних. У відповідь на ці виклики в галузі інформаційної безпеки сформувалися підходи до систематичного виявлення, класифікації та публікації інформації про вразливості.

Одним із ключових джерел для цього є Національна база даних вразливостей – NVD (англ. National Vulnerability Database), яка підтримується Національним інститутом стандартів і технологій США (NIST). У цій базі щороку публікується десятки тисяч нових вразливостей з унікальними ідентифікаторами CVE (Common

Vulnerabilities and Exposures), детальними описами, рівнями критичності, технічними характеристиками, прив’язкою до платформ і програмних продуктів. Станом на 2023 рік кількість опублікованих вразливостей перевищила 28 тисяч, що свідчить про безпрецедентний обсяг інформації, з якою мають працювати фахівці з кібербезпеки [1].

Однак просте існування такого джерела інформації не гарантує її ефективне використання. Більшість компаній або IT-відділів не мають змоги оперативно аналізувати велику кількість щоденних оновлень вручну. Крім того, значна частина вразливостей не стосується конкретної інфраструктури організації. Для ефективного реагування на потенційні загрози потрібен інструмент, який дозволяє автоматично виділяти лише ті записи, що відповідають встановленому набору продуктів, версій або платформ. У цьому контексті особливу роль відіграє стандарт CPE (Common Platform Enumeration) – формалізований механізм опису ідентичності програмного чи апаратного забезпечення, що дозволяє точно виявити, до яких саме компонентів стосується певна вразливість [2].

Таким чином, сформульована проблема полягає в необхідності створення системи, яка дозволяє:

- постійно моніторити базу даних NVD;
- автоматично фільтрувати вразливості відповідно до встановлених CPE-ідентифікаторів або ключових слів;
- своєчасно повідомляти відповідальних осіб про релевантні загрози через електронну пошту.

У межах цієї роботи предметом дослідження виступає процес автоматизованого виявлення та фільтрації вразливостей на основі відкритих структурованих джерел даних, а об’єктом – розроблена інформаційна система моніторингу, яка реалізує зазначену функціональність з використанням сучасних засобів веб-розробки та локального зберігання.

Наукова новизна запропонованої системи полягає у реалізації комплексного підходу до побудови інструменту моніторингу, що поєднує використання структурованих ідентифікаторів CPE для точної фільтрації, механізм

автоматичного сповіщення через електронну пошту, а також можливість роботи без зовнішніх залежностей на основі локальної бази даних. Такий підхід дозволяє інтегрувати систему в невеликі IT-середовища або локальні мережі без додаткових інвестицій у складні комерційні рішення.

Практична значущість розробки проявляється у здатності автоматизувати рутинні процеси аналізу ризиків і зробити реагування на потенційні загрози більш швидким і точним. За рахунок підтримки налаштувань отримувачів, списків моніторингів і інтервалів перевірки, система може гнучко адаптуватися до потреб окремих користувачів або організацій. Використання відкритого API та стандартних технологій забезпечує масштабованість і подальшу модернізацію.

1.2 Моніторинг вразливостей: терміни, стандарти та джерела даних

Моніторинг вразливостей є одним із ключових напрямів у сфері забезпечення інформаційної безпеки. Його основна мета полягає у виявленні, відстеженні та оцінюванні відомостей про вразливості в програмному забезпеченні з метою запобігання несанкціонованим доступам, атакам або порушенням цілісності систем. Важливою характеристикою ефективного моніторингу є своєчасність, структурованість і релевантність отриманої інформації щодо конкретних програмних компонентів.

У міжнародній практиці основним джерелом структурованої інформації про вразливості є Національна база даних вразливостей (NVD), що підтримується Національним інститутом стандартів і технологій США (NIST). NVD функціонує як відкрите репозиторій знань про вразливості, кожна з яких має унікальний ідентифікатор CVE (Common Vulnerabilities and Exposures). Ці ідентифікатори присвоюються за централізованою схемою та забезпечують єдину точку посилання на конкретну вразливість у різних інформаційних системах [3].

Усі записи CVE в базі NVD супроводжуються технічними характеристиками, серед яких дата публікації, опис проблеми, рівень критичності за шкалою CVSS (Common Vulnerability Scoring System), список заторкнутих продуктів, а також

посилання на пов’язані ресурси. Для програмної обробки цих даних база доступна у форматі JSON через офіційний API або у вигляді щоденних дампів, які містять актуальні записи. Важливо, що крім загальної інформації про вразливість, кожен запис містить прив’язку до відповідних CPE-записів, що дозволяє чітко встановити, які продукти потенційно зачеплені [4].

CPE (Common Platform Enumeration) – це стандартизований формат опису ідентичності апаратного чи програмного забезпечення. Він розроблений і підтримується в межах проєкту SCAP (Security Content Automation Protocol) і служить для точного позначення назв, версій, мов інтерфейсу та інших параметрів ПЗ. Формат CPE 2.3 представляє запис як послідовність частин, розділених двокрапками: наприклад, cpe:2.3:a:microsoft:edge:112.0.1722.64:*:*:*:*:*:*:* [5]. Така структура дозволяє однозначно інтерпретувати, про яке саме програмне забезпечення йде мова, включно з постачальником, версією, платформою тощо. Це дає змогу автоматизованим системам точно співвідносити вразливості з реальними конфігураціями користувачів.

Для виконання фільтрації вразливостей на основі CPE-сумісності необхідно розпізнати, чи входить заданий CPE до переліку заторкнутих продуктів у записі CVE. Цей перелік зазвичай міститься у полі configurations кожного JSON-документа NVD, де представлені логічні правила (наприклад, OR/AND-блоки), що описують залежності між продуктами. Обробка цих правил потребує парсингу та логічної інтерпретації в коді системи. Наприклад, одна вразливість може бути релевантною лише за наявності одночасно двох певних компонентів або певної комбінації версій, що ускладнює просту перевірку відповідності.

Додатково до CPE, деякі системи моніторингу дозволяють фільтрацію за ключовими словами в описі CVE або в полях, які вказують на використані технології, бібліотеки або функціональні блоки. Це особливо корисно у випадках, коли точний CPE не відомий, але відомі загальні характеристики або призначення вразливості.

Таким чином, ефективний моніторинг вразливостей передбачає глибоке розуміння структури NVD, особливостей формату CVE, логіки прив’язки до CPE

та механізмів обробки пов’язаних даних. Ці знання створюють основу для побудови автоматизованих систем, здатних забезпечувати точне і швидке виявлення потенційно критичних загроз у контексті конкретного програмного середовища.

1.3 Аналіз існуючих систем та методів моніторингу вразливостей

На сьогоднішній день існує низка інструментів та сервісів, призначених для моніторингу вразливостей, аналізу інформації з бази NVD та оперативного інформування про нові загрози. Найбільш доступним способом є безпосередній ручний пошук у базі NVD через її офіційний вебінтерфейс. Користувач може виконати запит за ключовими словами, фільтрувати за роком, рівнем критичності чи постачальником, однак така взаємодія потребує постійної участі людини та не дозволяє автоматизувати перевірку нових вразливостей для конкретного середовища [3]. Крім того, вебінтерфейс не підтримує фільтрацію за списком CPE-ідентифікаторів у зручному вигляді, що унеможливлює точне визначення релевантності кожної нової вразливості без ручного аналізу її конфігурацій.

Серед більш автоматизованих рішень варто виділити сервіс Vulners, який поєднує декілька джерел, включно з NVD, Red Hat, Ubuntu, Microsoft та іншими. Vulners API дозволяє виконувати запити до централізованої бази з уніфікованою структурою даних та підтримує пошук за CPE або CVE, однак сервіс є частково комерційним. Безплатна версія має обмеження щодо кількості запитів на добу, а також вимагає реєстрації та роботи через їхню інфраструктуру, що може бути небажаним у середовищах з підвищеними вимогами до конфіденційності [6].

Ще одним інструментом є сервіс Snyk, який спеціалізується на виявленні вразливостей у залежностях програмних проєктів. Він інтегрується з системами керування пакетами та може аналізувати репозиторії на GitHub або GitLab. Проте Snyk орієнтований насамперед на розробників програмного забезпечення, а не на універсальний моніторинг усіх компонентів інфраструктури. До того ж механізми

налаштування фільтрації є жорстко прив’язаними до структури проєкту, а не до зовнішньо визначених CPE, що обмежує гнучкість використання [7].

Варто також згадати про OpenVAS – систему для активного сканування вразливостей, яка використовує власну базу сигнатур і може проводити перевірку в режимі реального часу. Однак OpenVAS передбачає розгортання повноцінного сервера, потребує глибокої конфігурації та орієнтований на внутрішнє сканування мережі, а не на пасивний моніторинг баз публічних вразливостей.

На відміну від вищезгаданих інструментів, система, що розробляється у межах цієї роботи, має декілька принципових переваг. Вона не потребує складного розгортання чи інтеграції зі сторонніми сервісами. Весь процес реалізовано на основі відкритих стандартів і доступу до офіційного JSON-інтерфейсу NVD [4], без використання комерційних API або SDK. Завдяки фільтрації за повними CPE-ідентифікаторами та підтримці додаткових ключових слів, користувач отримує можливість максимально точно контролювати релевантність отриманих сповіщень. Крім того, система дозволяє формувати незалежні списки моніторингів для різних користувачів, що є рідкістю серед наявних сервісів з аналогічною функціональністю.

Водночас варто визнати, що спеціалізовані платформи, такі як Vulners або Snyk, мають більшу інфраструктурну підтримку, глибоку інтеграцію з CI/CD-процесами або репозиторіями, а також додаткові функції на кшталт візуального представлення аналітики. Натомість система, розроблена в межах цієї роботи, націлена на простоту, автономність, персоналізацію фільтрації та мінімальну залежність від зовнішніх компонентів, що є критично важливим у деяких категоріях застосування – наприклад, у внутрішніх корпоративних середовищах, які не допускають обміну даними з публічними платформами.

Таким чином, виконаний аналіз демонструє, що розроблене рішення займає проміжне положення між простими ручними інструментами пошуку і складними комерційними сервісами, заповнюючи нішу доступного, автоматизованого та налаштованого під користувача інструменту моніторингу вразливостей з бази NVD.

1.4 Вимоги до системи

На основі аналізу предметної області, дослідження доступних джерел даних і принципів функціонування існуючих рішень, сформульовано загальні вимоги до інформаційної системи моніторингу вразливостей. Визначення вимог дозволяє чітко окреслити функціональні можливості програмного продукту, а також встановити обмеження та очікувану поведінку з нефункціонального боку. У цьому підрозділі буде описано вимоги, які стосуються як зовнішнього вигляду та взаємодії користувача з системою, так і внутрішніх характеристик, що забезпечують її надійність, продуктивність, зручність і безпеку.

1.4.1 Нефункціональні вимоги системи

Нефункціональні вимоги визначають загальні характеристики системи, які не залежать від її функціональності, але безпосередньо впливають на якість, стабільність та зручність її використання. Вони охоплюють аспекти надійності, продуктивності, безпеки, зручності обслуговування та здатності до масштабування. Серед нефункціональних вимог даної слід виділити:

1. Надійність та стійкість до збоїв

Система повинна працювати стабільно протягом тривалого часу без зависань або втрати даних. У разі нештатних ситуацій вона має зберігати свій стан, і підтримувати можливість відновлення роботи без втручання користувача.

2. Продуктивність при великому обсязі даних

Щоденні оновлення бази NVD можуть містити тисячі записів, тому система повинна забезпечувати їх обробку впродовж обмеженого часу без помітного падіння швидкодії. Зокрема, обробка конфігурацій CPE та логіки фільтрації повинні бути оптимізовані для пакетної або асинхронної обробки [8].

3. Масштабованість та незалежність від інфраструктури

Рішення має бути придатним для розгортання як у локальних середовищах, так і в хмарних. Система повинна мати просту структуру без потреби у складному

програмному стеку, а всі залежності мають бути чітко задокументовані. Це дає змогу легко мігрувати її між різними середовищами.

4. Простота інсталяції та налаштування

Користувач повинен мати змогу швидко запустити систему з мінімальними налаштуваннями. Для цього застосовується SQLite як база даних, що не потребує встановлення серверної частини і може використовуватись у вигляді окремого файлу [4]. Налаштування системи повинні виконуватись через конфігураційний файл або змінні середовища.

5. Інтуїтивно зрозумілий інтерфейс

Якщо реалізується вебінтерфейс або конфігураційна панель, вона має бути простою, зрозумілою, без зайвого навантаження на користувача. Основні функції – додавання моніторингів, перегляд історії, редагування списку отримувачів – мають бути доступні в кілька дій.

6. Безпечна робота з обліковими даними

Під час надсилання email-сповіщень система має використовувати захищене з'єднання з SMTP-сервером, зокрема через TLS. Зберігання облікових даних для пошти повинно відбуватись через пароль додатку або інший безпечний механізм, наприклад, змінні середовища, щоб запобігти витоку критичної інформації [9].

7. Автономність роботи

Система повинна функціонувати повністю автоматично, виконуючи перевірки згідно з заданим графіком, без потреби у ручному запуску. Це включає автоматичне отримання нових вразливостей, фільтрацію відповідно до заданих правил і надсилання сповіщень без втручання адміністратора.

Сформульовані нефункціональні вимоги визначають ключові якісні характеристики системи, зокрема її стабільність, продуктивність, автономність та безпечність. Дотримання цих вимог забезпечить ефективне та надійне функціонування системи в різних середовищах із мінімальними витратами на супровід.

1.4.2 Функціональні вимоги системи

Функціональні вимоги описують основні дії, які система повинна виконувати для досягнення поставленої мети. Вони визначають, які саме функції мають бути реалізовані, як користувач взаємодіє з системою, які операції можливі над даними, а також яка логіка закладена у виконання перевірок, фільтрацій та сповіщень. Сформульовані нижче вимоги відображають ключовий функціонал системи моніторингу вразливостей.

1. Додавання нового моніторингу вразливостей

Система повинна забезпечувати можливість створення нового моніторингу, під час якого користувач вказує один або кілька ідентифікаторів CPE, а також, за потреби, ключові слова для фільтрації описів вразливостей. Кожен моніторинг повинен мати унікальну назву, періодичність перевірки та список основних налаштувань.

2. Збереження списку CPE та ключових слів для кожного моніторингу

Для кожного створеного моніторингу система повинна зберігати перелік CPE-записів, що описують цільові продукти, а також список додаткових ключових слів, які дозволяють уточнити фокус перевірки. Це забезпечує можливість точного контролю над тим, які вразливості вважаються релевантними для конкретного моніторингу.

3. Отримання та оновлення бази вразливостей з NVD

Система повинна регулярно звертатися до відкритих JSON-ресурсів NVD та отримувати актуальну інформацію про нові вразливості. Повинна бути реалізована можливість оновлення даних у локальному сховищі на основі дати останньої синхронізації, що дозволить уникати дублювання та зайвого навантаження [4].

4. Фільтрація нових вразливостей за заданими критеріями

Після отримання нових даних система повинна порівнювати кожну вразливість із заданими CPE та ключовими словами кожного моніторингу. Вразливості, що задовольняють хоча б одній з умов (відповідність CPE або виявлення ключового слова в описі), повинні вважатись релевантними.

5. Формування списку результатів перевірки

За результатами кожної перевірки система повинна формувати список знайдених вразливостей, які відповідають заданим критеріям. Цей список повинен бути збережений у локальній базі даних із вказанням дати, моніторингу, до якого належать результати, а також переліком CVE-ідентифікаторів, їхніх описів, рівня критичності та посилань на джерело.

6. Надсилання email-сповіщень про виявлені вразливості

У разі знаходження нових релевантних вразливостей, система повинна автоматично надсилати сповіщення на електронну адресу основного отримувача, вказаного в параметрах моніторингу. Повідомлення повинно містити стислий опис кожної знайденої вразливості та пряме посилання на її сторінку в NVD [3].

7. Налаштування списку додаткових отримувачів

Система повинна дозволяти для кожного моніторингу вказати додаткові email-адреси, які будуть отримувати ті самі повідомлення, що і основний користувач. Це забезпечить гнучке сповіщення кількох осіб у межах однієї організації.

8. Зміна налаштувань існуючих моніторингів

Користувач повинен мати можливість редагувати вже створені моніторинги, зокрема змінювати список CPE, ключові слова, отримувачів або періодичність перевірки. Це дає змогу адаптувати систему під зміну потреб користувача без створення нового моніторингу з нуля.

9. Перегляд історії перевірок та знайдених вразливостей

Інтерфейс або консольна частина системи повинна містити функціональність для перегляду результатів попередніх перевірок, в тому числі дати, кількості знайдених вразливостей, а також списку CVE з описами для кожного моніторингу.

10. Планування запуску перевірок за розкладом

Система повинна мати можливість самостійно запускати перевірки відповідно до встановленого інтервалу: наприклад, щоденно, щотижнево або з конкретною періодичністю у годинах. Такий запуск повинен бути реалізований

через вбудований планувальник або за допомогою інтеграції з cron або подібними системами.

11. Валідація введених даних під час створення моніторингів

Система повинна перевіряти коректність введених CPE-рядків, формат email-адрес та унікальність назв моніторингів, щоб уникнути помилок у роботі системи.

12. Ігнорування вже оброблених вразливостей

Система повинна зберігати інформацію про вже відправлені сповіщення, щоб не дублювати одні й ті самі вразливості в наступних звітах, навіть якщо вони все ще відповідають критеріям фільтрації.

Описані функціональні вимоги окреслюють основну логіку роботи системи, включно з механізмами додавання моніторингів, обробки даних з NVD, фільтрації, збереження результатів та сповіщення користувачів. Реалізація зазначених функцій дозволить створити повноцінне рішення для автоматизованого моніторингу вразливостей відповідно до поставлених цілей.

1.5 Діаграма варіантів використання

Для формалізації поведінки системи на початковому етапі проєктування доцільно використовувати уніфіковану мову моделювання UML. Одним із ключових елементів цієї мови є діаграма варіантів використання, яка дозволяє описати функціональну взаємодію між зовнішніми користувачами та системою. Така діаграма є інструментом, що допомагає візуалізувати основні сценарії використання, а також виявити ключові вимоги до інтерфейсу, поведінки та доступу до функцій системи.

Процес побудови діаграми починається з визначення акторів, тобто зовнішніх об'єктів, які ініціюють взаємодію з системою. У межах розроблюваної системи моніторингу вразливостей основним актором виступає Користувач, який виконує роль адміністратора налаштувань моніторингів. Користувач взаємодіє із системою через інтерфейс (графічний або консольний), встановлює критерії фільтрації, задає адреси для сповіщення та контролює результат виконання

перевірок. Крім того, у моделі розглядається Зовнішній сервіс NVD, який не є користувачем у класичному розумінні, але виконує роль джерела інформації для системи. Його включення до діаграми дозволяє візуально окреслити залежність системи від зовнішнього API. Повна діагарама варіантів використання системи представлена на рисунку 1.1.

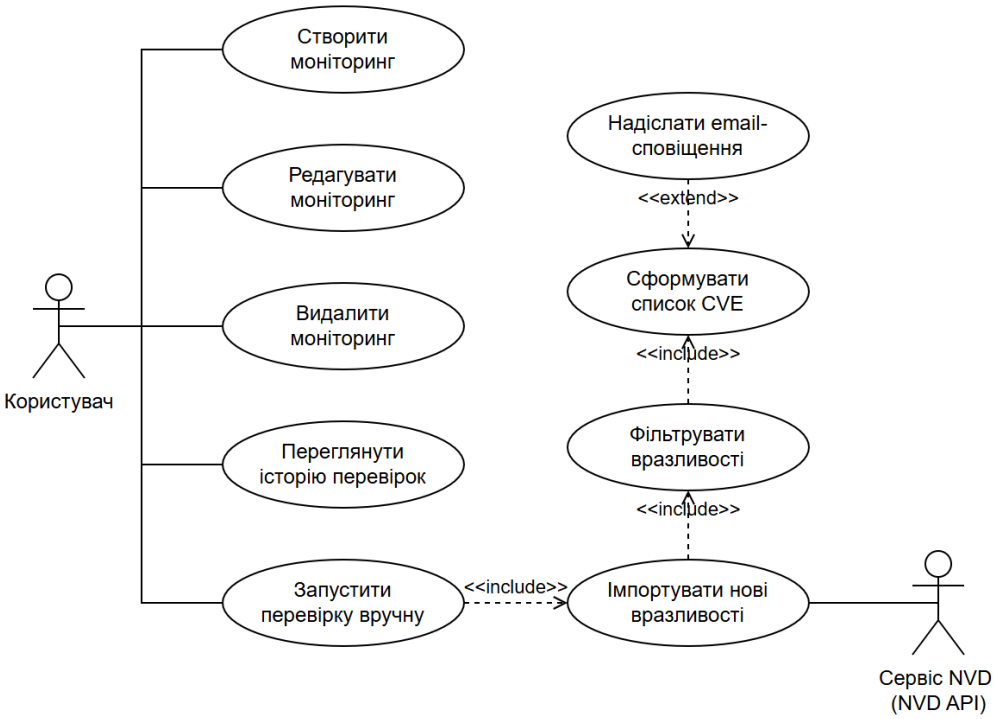

Рисунок 1.1 – Діаграма варіантів використання

Усі варіанти використання відображають ключові функціональні можливості, які були сформульовані в попередньому підрозділі. Діаграма варіантів використання відображає логіку взаємодії в зручній графічній формі та допомагає при подальшому розбитті функціоналу на програмні модулі.

2 ПРОЕКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ ВРАЗЛИВОСТЕЙ

Процес проєктування та реалізації є критичним етапом створення будь-якої програмної системи, оскільки саме на цьому етапі формуються ключові технічні рішення, визначається структура програмного продукту, його архітектура, внутрішня логіка та способи взаємодії компонентів. Якість проєктних рішень безпосередньо впливає на масштабованість, продуктивність, стабільність і здатність системи до подальшого розвитку. У цьому розділі представлено повну технічну концепцію створення системи моніторингу вразливостей: від архітектурної моделі до структури бази даних, розробки окремих програмних модулів та візуалізації логіки їх взаємодії. Також розглянуто засоби реалізації основної функціональності відповідно до вимог, сформульованих у попередньому розділі.

2.1 Архітектура та компоненти системи

Розроблена система моніторингу вразливостей має чітко визначену модульну архітектуру, що забезпечує логічне розділення відповідальностей між окремими частинами. Така побудова не лише спрощує підтримку та модифікацію коду, але й дозволяє повторно використовувати окремі модулі в інших проєктах або для розширення функціональності. Усі компоненти системи об’єднані навколо спільної мети – реалізації циклу автоматичного моніторингу, аналізу та інформування про нові релевантні вразливості.

Основу системи становить керуючий модуль, який координує виконання перевірок. Він відповідає за запуск усіх інших підсистем відповідно до встановленого інтервалу, зчитує налаштування моніторингів із бази даних і запускає послідовність операцій: завантаження нових даних, фільтрацію, збереження результатів та надсилання повідомлень. Завдяки використанню моделі

періодичних запусків, система працює автономно, не потребуючи ручного втручання на кожному кроці.

Модуль імпорту вразливостей взаємодіє з офіційними JSON-ресурсами NVD, отримуючи нові записи CVE через API або щоденні дампи. Для оптимізації трафіку реалізована перевірка дати останнього оновлення та завантаження лише тих файлів, які містять нові дані. Це дозволяє зменшити навантаження на зовнішній сервіс і підвищити ефективність роботи системи. Структура отриманих JSON-документів відповідає формату, що містить секції з заголовками, описами, метаданими, а також списками CPE та конфігураціями у вигляді логічних блоків [3].

Наступним етапом є модуль фільтрації, який аналізує кожен запис CVE та перевіряє його на відповідність CPE-ідентифікаторам і ключовим словам, що були задані у моніторингу. Логіка фільтрації реалізована як незалежний механізм, який може масштабуватись на багато моніторингів одночасно. Особливістю реалізації є можливість врахування складної конфігурації поля configurations у структурі NVD, що передбачає побудову дерева логічних умов OR/AND та вибірки цільових CPE.

Отримані результати передаються до сховища результатів, реалізованого на базі SQLite. У базі даних зберігаються всі моніторинги, їхні налаштування, списки отримувачів, результати перевірок, знайдені вразливості та технічна інформація про дату останньої синхронізації. Обраний тип СКБД забезпечує простоту використання, не потребує налаштування сервера, працює кросплатформено та дозволяє швидко створити резервну копію [4].

Ключовим з погляду взаємодії з користувачем є модуль сповіщень, який генерує email-листи для кожного моніторингу окремо, на основі нових виявлених вразливостей. Повідомлення формуються у форматі, зручному для читання, із включенням основної інформації про кожен CVE, рівень критичності та посилання на джерело. Надсилання здійснюється через SMTP-протокол із автентифікацією, для чого використовуються одноразові паролі додатків у межах Gmail або альтернативні механізми захищеного підключення [9].

Крім зазначених основних компонентів, система також включає допоміжні скрипти, які виконують адміністративні функції, зокрема ініціалізацію бази, тестовий запуск перевірок або скидання моніторингів до початкового стану. Таке доповнення дозволяє легше розгортати систему в новому середовищі, проводити оновлення або змінювати конфігурацію.

Архітектурна модель розробленої системи є прикладом легко розширюваної та керованої структури, в якій компоненти взаємодіють через чітко визначені інтерфейси, що мінімізує ризики помилок при зміні окремих частин. Це дозволяє в майбутньому легко інтегрувати нові джерела вразливостей, додати підтримку інших каналів сповіщення або запровадити більш складні алгоритми фільтрації.

2.2 Проєктування бази даних

Формування правильної структури бази даних є ключовим етапом у створенні інформаційної системи, яка покликана функціонувати стабільно, безпечно та ефективно в умовах обмежених ресурсів. База даних у системі моніторингу вразливостей виконує функцію централізованого сховища, що забезпечує збереження всіх налаштувань моніторингів, основних і додаткових отримувачів повідомлень, а також параметрів, необхідних для подальшого виконання перевірок. Її логічна структура проєктувалась з урахуванням простоти обслуговування, автономності роботи та мінімізації зайвих зв’язків, які могли б ускладнити реалізацію.

Зважаючи на відсутність потреби у багатокористувацькому доступі, клієнт-серверній взаємодії або складному адмініструванні, для збереження даних було обрано систему керування базами даних SQLite. Це вбудоване реляційне рішення, що не потребує окремого серверного оточення і забезпечує високу продуктивність для невеликих і середніх обсягів даних. Перевагою SQLite є простота інтеграції з Python-додатками, відсутність зовнішніх залежностей та можливість створення резервної копії у вигляді одного файлу бази [10].

На початковому етапі розробки було реалізовано дві ключові таблиці, які утворюють основу всієї логіки збереження даних. Центральне місце в структурі займає таблиця моніторингів, яка об'єднує в собі інформацію про об’єкт спостереження, критерії фільтрації, частоту перевірок та контактну інформацію. У цій таблиці для кожного моніторингу зберігається його назва, унікальний ідентифікатор, основний отримувач (email-адреса), задані CPE-ідентифікатори, ключові слова та інтервал, з якою має запускатися перевірка. Також фіксується дата останнього запуску, що дозволяє контролювати періодичність звернення до бази NVD та не пропускати критичних оновлень.

Дані CPE та ключові слова були реалізовані у вигляді полів з типом TEXT, що дозволило зберігати кілька значень у серіалізованому форматі (наприклад, JSON або розділені роздільниками). Такий підхід дає змогу уникнути зайвого ускладнення структури на початковому етапі розробки, а також забезпечити достатню гнучкість для обробки даних під час фільтрації. Водночас це рішення залишається відкритим для вдосконалення – у майбутньому окремі значення можуть бути винесені в допоміжні таблиці для точнішої нормалізації.

Другий компонент структури бази – це таблиця додаткових отримувачів. Вона зберігає електронні адреси тих користувачів, які також мають отримувати повідомлення про знайдені вразливості в рамках певного моніторингу. Зв’язок між записами встановлюється за допомогою зовнішнього ключа на основну таблицю моніторингів. Така структура дозволяє кожному моніторингу мати будь-яку кількість додаткових адрес, які розширюють канал інформування і підвищують надійність доставки сповіщень у разі відсутності основного користувача.

На рисунку 2.1 подано візуалізацію структури бази даних у SQLite після ініціалізації всіх необхідних таблиць.

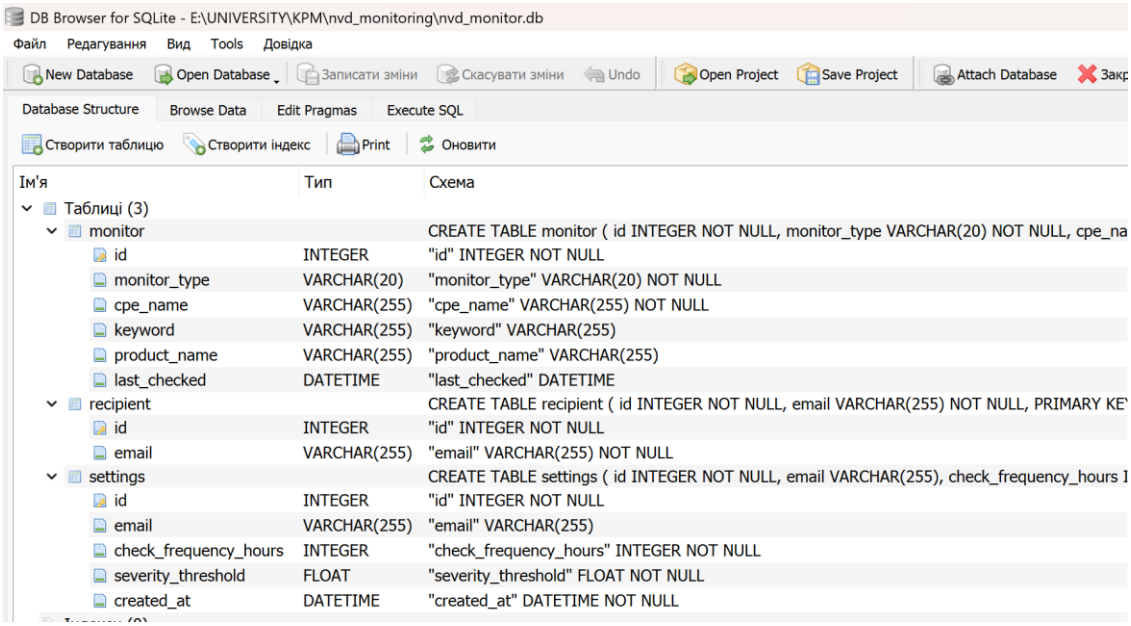

Рисунок 2.1 – Структура таблиць бази даних у SQLite

Фізичне створення таблиць здійснювалося за допомогою SQL-запитів, що містили інструкції для визначення типів даних, первинних та зовнішніх ключів, а також обмежень цілісності. Особлива увага приділялась забезпеченню правильного форматування полів, які мають зберігати email-адреси, щоб уникнути помилок при подальшому надсиланні повідомлень. Було передбачено автоматичну ініціалізацію бази при першому запуску системи, що спрощує процес її розгортання.

Описана структура є оптимальною для обраної моделі використання. Вона забезпечує достатній рівень гнучкості для роботи з моніторингами, дозволяє масштабувати систему шляхом додавання нових моніторингів без потреби змін у базі, а також зберігає простоту, необхідну для ефективної роботи у середовищах з обмеженими ресурсами. Такий підхід дозволяє поєднати швидкість розробки з можливістю подальшого розвитку архітектури відповідно до зростання вимог.

2.3 Діаграма класів

У процесі проєктування об’єктно орієнтованих систем UML діаграма класів є одним з основних інструментів опису статичної структури програмного продукту. Вона відображає класи системи, їх атрибути та операції, а також різноманітні види

зв’язків між ними. Такий підхід дає змогу отримати наочне уявлення про архітектуру програмного забезпечення, виявити основні абстракції та зрозуміти роль кожного компонента у загальній структурі рішення. У навчальній та інженерній літературі діаграму класів визначають як основу документації програмного проєкту, яка дозволяє моделювати об’єкти, їхні властивості й відносини у вигляді графічної схеми з класами та асоціаціями між ними [13,14]. Така діаграма особливо корисна для складніших систем, де текстовий опис архітектури є важким для сприйняття, а графічне подання дає змогу швидко охопити загальну структуру і глибше зрозуміти взаємодію компонентів [15].

Для розробленої системи моніторингу вразливостей з використанням NVD було побудовано UML діаграму класів, яка відображає основні сутності, пов’язані з налаштуванням моніторингів, отриманням та обробкою даних про вразливості, їх фільтрацією і надсиланням сповіщень. На рисунку 2.2 наведено діаграму класів, що візуалізує внутрішню структуру системи, зв’язки між класами та напрямок взаємодії між ними. Ця діаграма логічно відповідає структурі бази даних та модульній організації коду, описаним у попередніх підрозділах, і відображає ті групи функцій, які були реалізовані у програмному коді на Python з використанням Flask та SQLite.

Центральне місце на діаграмі посідає клас Monitoring. Він представляє окреме завдання моніторингу вразливостей і містить усі основні параметри, необхідні для його виконання. До атрибутів цього класу належать ідентифікатор моніторингу, назва, інтервал перевірки в годинах, список цільових CPE ідентифікаторів, перелік ключових слів, а також основна електронна адреса, на яку надсилаються результати. Також у класі передбачено зберігання дати останньої успішної перевірки, що дозволяє визначати, чи настав час для нового звернення до бази NVD. Методи класу Monitoring відповідають за оновлення його параметрів, ініціацію процесу перевірки, взаємодію з клієнтом NVD та модулем фільтрації, а також за формування структури даних, яка передається далі в модуль сповіщень. Клас Monitoring пов’язаний асоціацією з іншими класами системи і є логічним центром, що координує повний цикл моніторингу.

Для підтримки можливості надсилання результатів кільком користувачам у системі використовується клас AdditionalRecipient. Об’єкти цього класу зберігають електронні адреси додаткових отримувачів та посилання на відповідний екземпляр класу Monitoring. Такий зв’язок на діаграмі класів реалізований як асоціація типу один до багатьох, де один моніторинг може мати довільну кількість додаткових отримувачів. Клас AdditionalRecipient містить атрибут адреси електронної пошти, а також, за потреби, службові поля для ідентифікації запису у базі даних. Основні методи цього класу забезпечують створення, редагування та видалення додаткових адрес, а також валідацію коректності введених email значень. Клас не взаємодіє безпосередньо з NVD чи процесом фільтрації, проте відіграє важливу роль на етапі формування списку одержувачів у модулі надсилання повідомлень.

Взаємодія з зовнішнім джерелом інформації про вразливості реалізована за допомогою класу NvdApiClient. Цей клас інкапсулює логіку звернення до API або JSON фідів NVD, що є офіційним репозиторієм стандартизованих даних про вразливості, які базуються на ідентифікаторах CVE [16]. Клас NvdApiClient містить атрибути, пов’язані з параметрами підключення, такими як базова адреса сервісу, можливий API ключ та налаштування тайм-аутів. Його методи реалізують формування запитів до сервісу з урахуванням заданих фільтрів, зокрема CPE та часових інтервалів, а також отримання й попереднє перетворення JSON відповіді у внутрішні об’єкти. Саме цей клас використовується Monitoring для отримання списку актуальних вразливостей, що потенційно стосуються заданого продукту або платформи.

Отримані від NVD дані не використовуються безпосередньо, а відображаються на модельні об’єкти класу CveEntry. Цей клас представляє окрему вразливість і містить атрибути, що описують її сутність. Серед них ідентифікатор CVE, короткий текстовий опис, оцінка критичності, дата публікації, посилання на детальнішу інформацію та перелік CPE елементів, на які поширюється дана вразливість. Об’єкти CveEntry створюються і заповнюються класом NvdApiClient після парсингу отриманого JSON. Надалі вони передаються до класу фільтрації для відбору релевантних результатів. Завдяки наявності такого модельного класу

система працює з вразливостями як зі структурованими об’єктами, а не з сирими рядками JSON, що спрощує будь-які подальші операції над ними.

Логіку відбору релевантних записів реалізує клас VulnerabilityFilter. Він отримує колекцію об’єктів CveEntry разом з параметрами моніторингу, які містяться у класі Monitoring, зокрема цільовими CPE та списком ключових слів. У своїй роботі VulnerabilityFilter послідовно проходить отримані об’єкти, перевіряє відповідність кожної вразливості хоча б одному з вказаних CPE та аналізує вміст опису або інших текстових полів на наявність заданих ключових слів. Результатом роботи методу фільтрації є підмножина об’єктів CveEntry, які відповідають умовам моніторингу і мають бути включені до звіту та сповіщення. Цей клас не взаємодіє безпосередньо із зовнішніми сервісами або базою даних, проте є ключовою ланкою, що перетворює загальний список вразливостей на вузько таргетований набір, релевантний конкретному середовищу користувача.

Клас EmailNotifier відповідає за формування та надсилання email сповіщень за результатами моніторингу. У ньому зосереджені всі налаштування, пов’язані з використанням поштового сервера, зокрема адреса SMTP сервера, порт, ознака використання захищеного з’єднання, логін облікового запису та пароль додатка, який застосовується для автентифікації у поштовій службі. На практиці в системі використовується окремий поштовий обліковий запис у сервісі Gmail з паролем додатка, що відповідає сучасним вимогам безпеки до автентифікації зовнішніх застосунків. Основні методи класу EmailNotifier приймають об’єкт Monitoring, список релевантних CveEntry та перелік адрес отримувачів, сформований на основі основного email моніторингу та об’єктів класу AdditionalRecipient. Далі ці методи створюють текст повідомлення, включаючи заголовок, огляд знайдених вразливостей і посилання на детальні описи, після чого здійснюють надсилання листа через SMTP. Якщо сповіщення успішно відправлене, клас може повертати інформацію про статус, яка потім буде записана до журналу подій.

Для організації автоматичного циклічного запуску перевірок у системі використовується допоміжний клас MonitoringScheduler. У загальному випадку він може працювати поверх засобів мови Python для реалізації періодичних завдань і

містить посилання на колекцію об’єктів Monitoring. Основне його завдання полягає у тому, щоб через певні проміжки часу перевіряти, які моніторинги досягли чергового моменту виконання, та ініціювати для них повний цикл перевірки, що включає виклики NvdApiClient, VulnerabilityFilter та EmailNotifier. Наявність такого класу дозволяє відокремити логіку планування від решти бізнес логіки застосунку та забезпечити можливість централізованого керування всіма циклічними завданнями.

Ще один важливий допоміжний компонент, який відображено на діаграмі, це клас AppLogger. Він відповідає за фіксацію подій, що відбуваються під час роботи системи: успішне отримання даних від NVD, кількість знайдених релевантних вразливостей, статус відправлення сповіщень, а також різноманітні помилки, наприклад проблеми підключення до мережі або поштового сервера. Через методи AppLogger різні класи, зокрема Monitoring, NvdApiClient та EmailNotifier, можуть записувати текстові повідомлення до лог файлу або іншого сховища журналу. Це полегшує діагностику та супровід системи, а також створює додаткову доказову базу для аналізу коректності роботи інструменту моніторингу.

Усі описані класи утворюють узгоджену модель, що відображена на UML діаграмі класів на рисунку 2.2. Клас Monitoring пов’язує між собою рівень конфігурації моніторингів, обробку вразливостей та сповіщення, NvdApiClient і CveEntry представляють шар інтеграції з базою NVD та модель доменної сутності вразливості, VulnerabilityFilter відповідає за прикладну логіку відбору релевантних записів, EmailNotifier, AdditionalRecipient, MonitoringScheduler та AppLogger забезпечують допоміжні сервісні функції, необхідні для повноцінного автоматичного моніторингу.

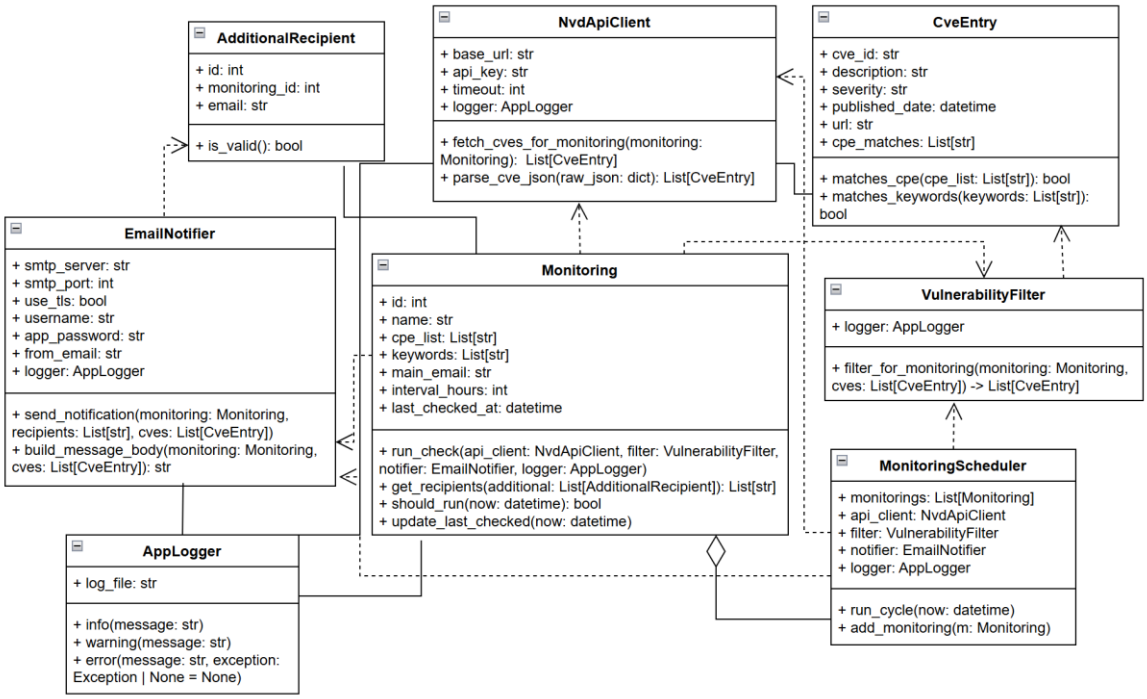

Рисунок 2.2 – Діаграма класів системи

Узагальнюючи, діаграма класів системи моніторингу вразливостей відображає цілісну внутрішню структуру розробленого програмного забезпечення та взаємодію між його основними складовими. Вона демонструє, як класи моніторингу, імпорту даних, фільтрації, сповіщень та допоміжні сервіси узгоджено працюють разом для реалізації повного циклу виявлення та повідомлення про вразливості. Така модель не лише формалізує поточну реалізацію системи, а й створює надійну основу для її подальшої модернізації, масштабування та супроводу.

2.4 Реалізація основних модулів системи

Реалізація основних модулів є критичним етапом у створенні системи моніторингу вразливостей, оскільки саме на цьому рівні абстрактна архітектура перетворюється на конкретний програмний механізм, здатний працювати з реальними даними. Від того, наскільки коректно та продумано реалізовані модулі отримання даних, їх обробки й сповіщення користувачів, безпосередньо залежать

надійність роботи системи, своєчасність виявлення вразливостей та зручність використання інструменту на практиці. У цьому підрозділі розглядається програмна реалізація ключових компонентів системи, зокрема модуля інтеграції з NVD, який забезпечує доступ до актуальної інформації про вразливості і є відправною точкою для всього подальшого процесу моніторингу.

2.4.1 Модуль отримання даних з NVD

Модуль отримання даних з NVD є одним з ключових елементів розробленої системи, оскільки саме він забезпечує доступ до актуальної інформації про вразливості, опубліковані в Національній базі вразливостей. NVD надає можливість програмного доступу до даних через JSON фіди та REST API версії 2.0, які повертають уніфікований набір відомостей про вразливості у форматі JSON з підтримкою фільтрації за різними параметрами, зокрема за CPE ідентифікаторами та часовими діапазонами [17, 18].

У розробленій системі модуль взаємодії з NVD реалізовано у вигляді окремого сервісного класу NvdApiClient, який інкапсулює всі деталі побудови HTTP запитів, обробки відповіді та перетворення отриманих даних у внутрішні об’єкти CveEntry.

Для організації запитів до NVD використано бібліотеку requests, яка є де-факто стандартом для виконання HTTP запитів у Python і надає зручний високорівневий інтерфейс для роботи з REST сервісами [19].

Клас NvdApiClient зберігає базову адресу API NVD у вигляді рядка, наприклад https://services.nvd.nist.gov/rest/json/cves/2.0, а також параметри тайм-ауту запиту та, за потреби, API ключ, що використовується для обходу обмежень швидкості запитів. Під час виконання чергового моніторингу метод модуля формує запит з відповідними параметрами, які залежать від налаштувань обраного моніторингу. Основними з них є CPE, що задається параметром cpeName, а також, за наявності відповідної логіки у системі, часові обмеження публікації або модифікації вразливостей, які дають змогу отримувати лише останні зміни.

У загальному випадку запит до API NVD для одного моніторингу має вигляд HTTP GET звернення до ресурсу cves/2.0 з переданням параметрів у рядку запиту. Модуль використовує можливості фільтрації, які описані у специфікації API NVD, зокрема параметр cpeName, що дозволяє отримати всі вразливості, пов’язані з конкретним програмним або апаратним продуктом у форматі CPE 2.3 [17].

Водночас враховується факт, що NVD застосовує пагінацію, тому результати повертаються сторінками з обмеженим значенням resultsPerPage і можливістю зсуву через параметр startIndex. У реалізованому модулі обрано таку стратегію, за якої для щоденного моніторингу достатньо одного або кількох послідовних запитів із розумним обмеженням кількості записів, що дозволяє уникати надмірного навантаження на зовнішній сервіс і водночас отримувати повний перелік релевантних записів для конкретного моніторингу.

Приклад спрощеного фрагмента коду, який демонструє загальний підхід до отримання даних, представлений в лістингу 2.1.

Лістинг 2.1 – Фрагмент демонстрації загального підходу до отримання даних

```
response = requests.get(
    "https://services.nvd.nist.gov/rest/json/cves/2.0",
    params={
        "cpeName": cpe_value,
        "resultsPerPage": 200
    },
    timeout=10
)
response.raise_for_status()
raw_data = response.json()
```

У реальній реалізації цей фрагмент доповнюється обробкою помилок, логуванням, підтримкою кількох CPE для одного моніторингу та, за потреби, обходом сторінок результатів. Наведений приклад ілюструє принципову схему: клієнт формує HTTP запит, отримує JSON відповідь та передає її для подальшого аналізу і перетворення.

Структура відповіді NVD у форматі JSON 2.0 описана в офіційній документації API. Вона містить заголовкову частину з полями, що описують

формат, версію відповіді та момент її формування, а також основний масив даних про вразливості. У параметрі vulnerabilities повертається масив об’єктів, де кожен елемент містить вкладений об’єкт cve з ідентифікатором CVE, текстовим описом, посиланнями, метаданими та структурами, які відображають показники критичності за CVSS.

Також у відповіді присутня інформація про конфігурації, які визначають, до яких CPE належить конкретна вразливість. Саме ці дані надалі використовуються для формування списку CPE, що зберігаються у внутрішньому представленні CveEntry.

Отримані від NVD JSON дані перетворюються у внутрішні структури за допомогою стандартного модуля json мови Python, який забезпечує серіалізацію та десеріалізацію JSON у словники, списки та інші базові типи [20].

На першому етапі відповідь перетворюється у словник верхнього рівня, з якого виділяється масив vulnerabilities. Далі модуль проходить по кожному елементу цього масиву та зчитує ключові поля, які є необхідними для подальшої фільтрації та сповіщення, а саме ідентифікатор CVE, текстовий опис, дату публікації, значення критичності та перелік CPE, що зазначені у конфігураційних об’єктах. Для кожного такого запису створюється об’єкт класу CveEntry, атрибути якого заповнюються на основі відповідних полів JSON структури. У результаті модуль не працює з «сирим» JSON, а оперує вже типізованими об’єктами, зручними для обробки в інших частинах системи.

Важливим рішенням під час реалізації було не зберігати отримані від NVD вразливості у базі даних. Замість цього обробка списку CveEntry здійснюється в оперативній пам’яті в межах одного циклу моніторингу. Такий підхід дозволяє завжди працювати з найактуальнішою інформацією, зменшує обсяг локальних даних, що потребують захисту, і спрощує структуру бази даних, яка зосереджується на зберіганні налаштувань моніторингів та списків отримувачів, а не на копіюванні великої частини NVD. У разі потреби розширення системи модуль отримання даних може бути доповнений механізмами кешування, зберігання історії виявлених вразливостей та повторного використання вже завантажених

фрагментів, однак поточна реалізація орієнтована на простий і прозорий сценарій щоденного моніторингу.

Для підвищення надійності роботи модуль NvdApiClient інтегровано з механізмом логування, який фіксує ключові події, такі як успішні звернення до API, коди повернення HTTP, обсяг отриманих даних, а також помилки на кшталт недоступності сервісу або перевищення часу очікування. У разі виникнення помилок модуль повертає контроль викликачеві разом з деталями помилки, що дозволяє або повторити запит, або повідомити користувача про тимчасову недоступність зовнішнього джерела. Завдяки цьому модуль отримання даних з NVD перетворюється на стабільний і передбачуваний компонент, який забезпечує системі надійний доступ до центрального джерела інформації про вразливості.

Таким чином, модуль отримання даних з NVD реалізує повний цикл взаємодії із зовнішнім репозиторієм: від формування запиту з урахуванням налаштувань моніторингу, через отримання й розбір JSON відповіді, до побудови внутрішнього набору об’єктів CveEntry, які надалі обробляються модулем фільтрації та модулем сповіщень. Обрані технології та формат обміну даними відповідають сучасним рекомендаціям щодо використання NVD API 2.0 і забезпечують достатню гнучкість для подальшого розвитку системи.

2.4.2 Модуль фільтрації вразливостей за CPE

У модулі фільтрації вразливостей відбувається перехід від великого масиву сирих даних, отриманих з NVD, до вузько сфокусованого набору записів, що справді стосуються програмних продуктів, зазначених у налаштуваннях моніторингу. Саме на цьому етапі реалізується практичне використання стандарту Common Platform Enumeration як єдиного способу ідентифікації платформ, операційних систем та прикладного ПЗ, для яких описано вразливості [21, 22]. CPE слугує «спільною мовою» між NVD та системами моніторингу, а отже коректна реалізація модуля, що порівнює CPE в налаштуваннях моніторингу з CPE у записах CVE, є критично важливою для точності результатів відбору [23].

У розробленій системі логіка фільтрації реалізована у вигляді окремого сервісного класу VulnerabilityFilter, який працює з колекцією об’єктів CveEntry, отриманих модулем NvdApiClient. Кожен об’єкт CveEntry містить не лише ідентифікатор та опис вразливості, але й список CPE-рядків, що представляють продукти, яких вона стосується. Клас VulnerabilityFilter отримує також об’єкт Monitoring, з якого зчитуються налаштування CPE та, за потреби, ключові слова. Таким чином, модуль фільтрації не прив’язаний безпосередньо до способу отримання даних з NVD і працює на рівні внутрішньої моделі, відсікаючи усе зайве й залишаючи лише ті вразливості, які релевантні конкретному моніторингу.

Список CPE для кожного моніторингу задається у вигляді одного текстового поля у базі даних, де окремі значення розділяються символом-роздільником (наприклад комою або переносом рядка). При завантаженні налаштувань моніторингу із бази це поле розбивається на список рядків, кожен з яких відповідає одному CPE-ідентифікатору. Такі ідентифікатори використовують формат CPE 2.3, у якому рядок складається з послідовності дванадцяти компонентів, розділених двокрапками, і починається префіксом cpe:2.3 [22]. Стандарт визначає семантику кожного компонента, включно з типом сутності (додаток, операційна система чи апаратне забезпечення), виробником, назвою продукту, версією, оновленнями, мовною локалізацією та іншими деталями. У рамках даного проєкту ці рядки зберігаються як цілісні значення, без розбору на окремі поля в базі, що спрощує структуру сховища й достатньо для реалізації практичної фільтрації у пам’яті застосунку.

Алгоритм перевірки відповідності CPE побудований за принципом покрокового звуження множини вразливостей. Після того як модуль NvdApiClient сформував список об’єктів CveEntry для певного моніторингу, VulnerabilityFilter послідовно проходить кожен з них і аналізує його поле зі списком CPE-ідентифікаторів. Для кожного CveEntry модуль порівнює цей список із набором CPE, вказаних у налаштуваннях моніторингу. Найпростіший і надійний випадок передбачає повний збіг рядків CPE, коли вразливість явно позначена тим самим ідентифікатором, що й продукт у системі користувача. Однак з огляду на те, що в

практиці NVD часто застосовуються шаблони з використанням символів-заповнювачів, а також узагальнені назви продуктів, у модулі реалізовано перевірку на часткову відповідність, коли CPE моніторингу виступає як характерний префікс або підрядок у CPE, що міститься у записі CVE [21, 22].

З технічної точки зору порівняння реалізоване у вигляді подвійного циклу, який проходить усі CPE із запису CveEntry та всі CPE, задані у моніторингу. Для кожної пари значень виконується перевірка відповідності, яка, залежно від обраної логіки, може включати перевірку на повну рівність, збіг початку рядка або наявність CPE моніторингу як підрядка у значенні з NVD. За першої ж успішної перевірки вразливість позначається як «відповідна» за CPE й переходить до наступного етапу фільтрації, а якщо збігів не знайдено, запис відкидається. Фактично це означає, що у подальших кроках система працює лише з тими вразливостями, які належать до вузького кола продуктів, явно вказаних користувачем у налаштуваннях моніторингу, що істотно зменшує кількість шуму у результатах.

Окрім фільтрації за CPE, у розробленій системі передбачено додаткове обмеження за ключовими словами, яке застосовується до текстового опису вразливості й дає змогу ще точніше сфокусуватися на певних компонентах або контекстах використання продукту. Тим не менш саме CPE-фільтрація залишається основою відбору, оскільки прив’язує кожен запис CVE до конкретної платформи або програмного забезпечення, що використовується в інфраструктурі організації [21, 23]. Ключові слова виконують роль додаткового рівня уточнення, а не заміни стандартизованої ідентифікації, забезпечуючи поєднання формальних і неформальних критеріїв відбору.

З погляду продуктивності модуль фільтрації виконує операції лінійної складності відносно кількості отриманих вразливостей і кількості CPE, заданих у моніторингу. Зважаючи на те, що в рамках типової конфігурації система працює з результатами щоденних або періодичних запитів до NVD і розглядає лише обмежений підмножину CVE, пов’язану з конкретним продуктом, такий підхід є цілком достатнім за швидкодією. За потреби подальшого масштабування можливе

оптимізування алгоритму, наприклад попередня нормалізація CPE, побудова індексів або використання більш формальних процедур порівняння, описаних у специфікації CPE Name Matching [22].

У підсумку модуль фільтрації вразливостей за CPE реалізує ключову бізнес-логіку системи: на основі списку CPE, збереженого у налаштуваннях моніторингу, він відбирає з великого масиву NVD лише ті вразливості, які дійсно стосуються цільових продуктів. Завдяки використанню стандартизованої схеми CPE та поєднанню простих, але ефективних методів порівняння, цей компонент забезпечує високу релевантність результатів і робить систему придатною для практичного використання у задачах управління вразливостями.

2.4.3 Модуль сповіщення

Модуль сповіщення є завершальним елементом ланцюжка обробки даних у системі моніторингу вразливостей, адже саме він перетворює технічні результати фільтрації на зрозуміле повідомлення для фахівця з безпеки. Його завдання полягає у тому, щоб зібрати всі релевантні вразливості, сформувати на їх основі інформативний лист та надійно доставити його на вказані електронні адреси. Від якості реалізації цього модуля залежить, чи зможе користувач оперативно реагувати на нові загрози, не витрачаючи час на ручний перегляд бази NVD та ручне складання звітів.

У розробленій системі функціональність сповіщення реалізована у вигляді сервісного класу EmailNotifier, який взаємодіє з класами Monitoring, AdditionalRecipient та набором відфільтрованих об’єктів CveEntry. Після завершення роботи модуля отримання даних з NVD та модуля фільтрації, клас Monitoring передає до EmailNotifier список релевантних вразливостей, а також сформований список одержувачів. До цього списку входить основна адреса, збережена у полі моніторингу, та одна або кілька додаткових адрес із пов’язаних об’єктів AdditionalRecipient. Таким чином формується єдиний набір одержувачів,

для яких в межах одного моніторингу відправляється повідомлення про знайдені вразливості.

Частота відправлення листів визначається не модулем сповіщення, а налаштуваннями моніторингу та планувальником MonitoringScheduler. Кожен раз, коли для певного моніторингу виконується цикл перевірки, після отримання та фільтрації списку вразливостей модуль сповіщення викликається з актуальними даними. Якщо для цього запуску не знайдено жодної вразливості, що відповідає заданим CPE та ключовим словам, система може не формувати лист, щоб уникнути надмірної кількості порожніх сповіщень та зменшити шум у поштовій скриньці користувача. Якщо ж список не порожній, формується повноцінний звіт, який надсилається всім одержувачам, пов’язаним з моніторингом.

Формат сформованого повідомлення орієнтований на зручність швидкого перегляду та подальшого аналізу. У полі теми листа зазвичай зазначається назва моніторингу та коротка вказівка на факт виявлення нових вразливостей, наприклад назва продукту або кількість знайдених записів. У тексті листа наводиться загальний вступ, де вказується, для якого CPE та в який момент часу було виконано перевірку, а далі йде структурований перелік вразливостей. Для кожного об’єкта CveEntry виводиться ідентифікатор CVE, рівень критичності, дата публікації та короткий опис, а також посилання на повну інформацію на сайті NVD. Такий формат дозволяє одразу оцінити масштаб проблеми та швидко перейти до детальної інформації, якщо це необхідно. Повідомлення реалізовано у вигляді текстового листа, що забезпечує максимальну сумісність з різними поштовими клієнтами, а також спрощує автоматичну обробку та переадресацію. За потреби структура модуля сповіщення дозволяє розширити його підтримкою HTML або multipart-повідомлень, де одночасно надсилаються текстова та HTML версії листа, що відповідає сучасним рекомендаціям щодо поєднання зручності читання та доброї доставлюваності електронної пошти [27].

Для реалізації власне процесу відправлення використовується стандартний модуль smtplib мови Python, який забезпечує клієнтську реалізацію протоколу SMTP та дозволяє встановлювати з’єднання з поштовими серверами, виконувати

автентифікацію та надсилати сформовані повідомлення [24]. У якості транспортного рівня обрано SMTP сервер Gmail smtp.gmail.com з використанням порту 587 та захищеного з'єднання TLS, що відповідає офіційним рекомендаціям Google для надсилання листів із застосунків [25].

Для автентифікації застосовується не основний пароль облікового запису, а пароль додатка, створений у налаштуваннях безпеки облікового запису Google, що підвищує безпеку взаємодії і відповідає сучасним вимогам до захисту доступу до поштових сервісів [9,25]. Після встановлення з'єднання модуль виконує процедуру привітання із сервером, ініціює шифроване з'єднання, а потім виконує вхід із зазначеними обліковими даними і надсилає повідомлення з правильно сформованими SMTP конвертом і заголовками.

Формування самого листа здійснюється за допомогою стандартних засобів роботи з MIME-повідомленнями з бібліотеки email, що входить до складу стандартної бібліотеки Python [24]. У модулі EmailNotifier створюється об'єкт повідомлення, у якому задаються поля From, To та Subject, причому список одержувачів формується як кома-розділений рядок на основі сукупності основної та додаткових адрес. Далі в тіло листа додається сформований текстовий блок з інформацією про вразливості. На заключному етапі модуль перетворює об'єкт повідомлення у рядкове представлення та передає його в метод надсилання sendmail клієнта SMTP. Такий підхід відповідає типовим рекомендаціям щодо відправлення пошти з Python та дозволяє за потреби розширити лист вкладеннями або альтернативними форматами без кардинальної зміни коду [24, 26].

Особливу увагу в реалізації модуля сповіщення приділено коректній роботі зі списком одержувачів. Для уникнення дублювання листів модуль формує множину адрес, до якої спочатку додається основна адреса з моніторингу, а потім усі адреси з об'єктів AdditionalRecipient. Після цього множина перетворюється у впорядкований список, що використовується як у SMTP конверті, так і у заголовку To. Така схема дає змогу гарантувати, що кожен одержувач отримає рівно один екземпляр повідомлення, навіть якщо адреса випадково була вказана і як основна, і як додаткова. Аналогічний підхід рекомендовано для масових розсилок в

прикладних системах, де важливо коректно формувати заголовки повідомлень і уникати помилок, пов’язаних із некоректним переліченням одержувачів [26].

Надійність модуля сповіщення додатково підвищується завдяки інтеграції з механізмом логування AppLogger. У разі успішного надсилання листа до журналу вноситься запис із зазначенням моніторингу, часу відправлення та кількості адресатів. Якщо під час роботи виникає помилка, наприклад недоступність SMTP сервера, помилкові облікові дані або мережевий збій, модуль фіксує у журналі відповідний запис із текстом помилки, що дає змогу адміністраторам або розробникам оперативно діагностувати проблему. За потреби логіка може бути доповнена повторними спробами надсилання або локальним буферизаційним механізмом, однак навіть у базовій конфігурації модуль забезпечує прозорість та трасованість процесу сповіщення.

У підсумку модуль сповіщення реалізує завершальний крок у роботі системи моніторингу вразливостей: на основі відібраних модулем фільтрації записів він формує структурований текстовий звіт і безпечно доставляє його всім зацікавленим особам через захищене SMTP з використанням облікового запису Gmail. Така реалізація поєднує простоту використання, відповідність загальноприйнятим стандартам електронної пошти та достатню гнучкість для подальшого розвитку формату повідомлень і політики розсилки.

3 ВПРОВАДЖЕННЯ, ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

Масштабне розповсюдження вразливостей та постійна зміна ландшафту загроз роблять етап тестування, впровадження та підтримки системи моніторингу особливо критичним. Саме на цьому етапі перевіряється, наскільки надійно та стабільно працює реалізоване рішення в реальних умовах, чи коректно воно виявляє релевантні вразливості, своєчасно інформує відповідальних осіб та здатне безперервно функціонувати в інфраструктурі організації.

Важливість цього етапу полягає в тому, що навіть добре спроєктована та реалізована система не принесе очікуваного ефекту без коректного розгортання, налаштування розкладу перевірок, організації сповіщень і подальшого супроводу. Тестування, впровадження та підтримка дозволяють перетворити програмний прототип на практичний інструмент, який реально підсилює процес управління вразливостями та знижує ризики для інформаційної безпеки.

3.1 Впровадження системи

Впровадження системи моніторингу вразливостей починається з початкового налаштування середовища виконання та поетапного запуску всіх компонентів, що забезпечують повний цикл роботи: від конфігурації Flask-застосунку і бази даних до організації автоматичного щоденного запуску перевірок. Коректне проходження цих кроків є важливим, оскільки саме вони перетворюють набір вихідних файлів проєкту на готову до експлуатації систему, яку можна запускати на реальному робочому місці без додаткових доопрацювань.

Першим кроком було створено окреме віртуальне середовище Python для проєкту, щоб ізолювати залежності та уникнути конфліктів з іншими застосунками. Для цього використовувався стандартний модуль venv, рекомендований документацією Python для організації ізольованих середовищ [28]. Після створення каталогу віртуального середовища воно активувалося у командному рядку, а далі через pip встановлювалися всі необхідні пакети, зокрема Flask, бібліотеки для

роботи з HTTP-запитами та взаємодії з SQLite [28, 29]. Такий підхід забезпечив відтворюваність середовища: на будь-якій іншій машині можна виконати ті самі команди та отримати ідентичну конфігурацію.

Наступним кроком було створено найпростіший Flask-застосунок, який служив перевіркою коректності налаштування середовища. У файлі, що виконує роль точки входу, був описаний мінімальний маршрут, який повертає текстове повідомлення українською мовою.

Лістинг 3.1 – Перевірка налаштування середовища

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def index():
    return "Flask працює! Середовище налаштовано правильно."
```

Після запуску застосунку командою flask run у браузері відкривалася локальна адреса, де відображалося це повідомлення. Це підтверджувало, що Flask встановлено коректно, сервер працює, а середовище налаштовано без помилок відповідно до офіційного посібника з швидкого старту Flask [30]. На рисунку 3.1 доцільно розмістити скріншот вікна браузера з цим повідомленням, який фіксує завершення початкового етапу налаштування.

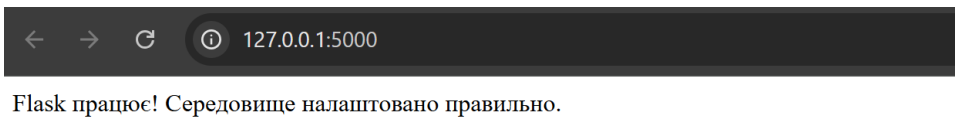

Рисунок 3.1 – Перевірка коректності налаштування середовища Flask

Після того, як було підтверджено працездатність базового застосунку, наступним етапом стало впровадження сторінки з налаштуваннями для основного та додаткових отримувачів, а також кнопки ручного запуску перевірки. На цьому етапі до проєкту додали підключення до бази даних SQLite, де зберігаються налаштування отримувачів, а у Flask були реалізовані маршрути для відображення сторінки з формою та обробки надсилання даних. У шаблоні HTML було сформовано поля для введення основної адреси електронної пошти, текстове поле для зазначення одного або кількох додаткових адрес, а також елемент керування для ініціації ручного запуску повного циклу моніторингу. Кнопка ручного запуску використовувалася на етапі впровадження та тестування для того, щоб переконатися, що всі модулі – отримання даних з NVD, фільтрація та сповіщення – коректно працюють разом, ще до налаштування автоматичного запуску за розкладом. Скріншот цієї сторінки з формою налаштувань і кнопкою ручного запуску можна розмістити як рисунок 3.2.

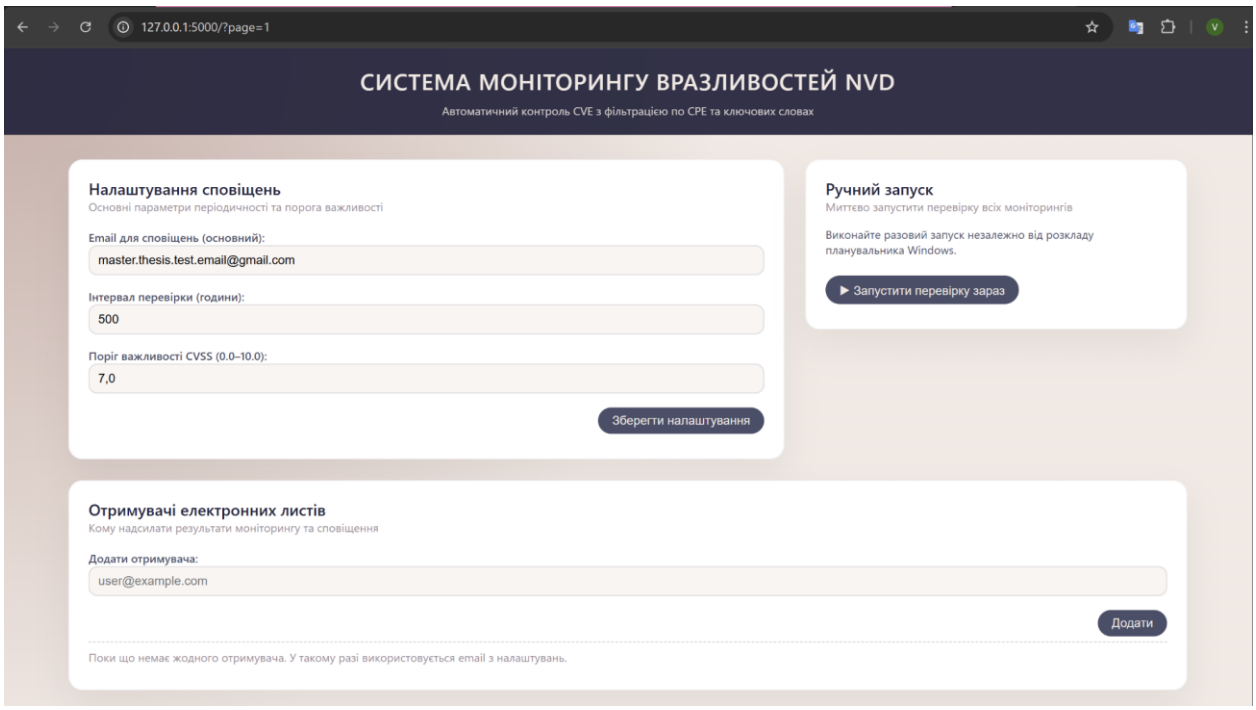

Рисунок 3.2 – Сторінка налаштування основного та додаткових отримувачів і ручного запуску перевірки

Далі було реалізовано інтерфейс для роботи зі списком моніторингів, який є центральним елементом конфігурації системи. У базі даних була створена таблиця для зберігання моніторингів, де для кожного запису фіксується назва моніторингу, тип моніторингу, список CPE або ключових слів, інтервал перевірки та інші параметри, необхідні для подальшої обробки. У веб-інтерфейсі з'явилася окрема сторінка, на якій у табличному форматі відображається перелік усіх налаштованих моніторингів. Для кожного моніторингу можна було задати назву продукту, вибрати, чи це моніторинг за CPE, чи за ключовим словом, та заповнити відповідні поля. Там же реалізовано можливості додавання нового моніторингу та видалення наявного, що дозволило гнучко керувати конфігураціями без ручного редагування бази даних. На цьому етапі було важливо переконатися, що всі зміни коректно зберігаються у SQLite та відображаються при повторному відкритті сторінки. Відповідний скріншот зі сторінкою, де показано список моніторингів з можливістю додавання та видалення, може бути доданий як рисунок 3.3.

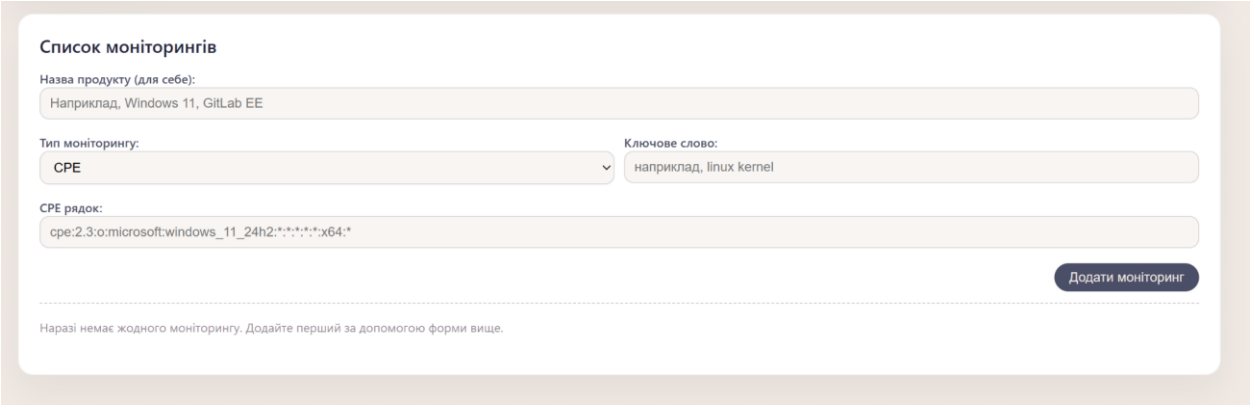

Рисунок 3.3 – Сторінка зі списком моніторингів та керуванням їх конфігурацією

Окремим кроком було впровадження можливості редагування вже створеного моніторингу. Для цього додатково налаштовано маршрут Flask, який приймає ідентифікатор моніторингу, завантажує його дані з бази та передає їх у шаблон HTML. У формі редагування поля попередньо заповнюються поточними значеннями, що дозволяє користувачу при необхідності змінити тип моніторингу, оновити список CPE або ключових слів, скоригувати інтервал перевірки чи змінити

назву. Після надсилання форми оновлені дані знову записуються у базу SQLite, а користувач повертається до сторінки зі списком моніторингів. Такий сценарій істотно підвищує зручність експлуатації системи, оскільки конфігурацію можна гнучко змінювати без необхідності створювати моніторинг заново. На рисунку 3.4 доцільно розмістити скріншот сторінки редагування моніторингу.

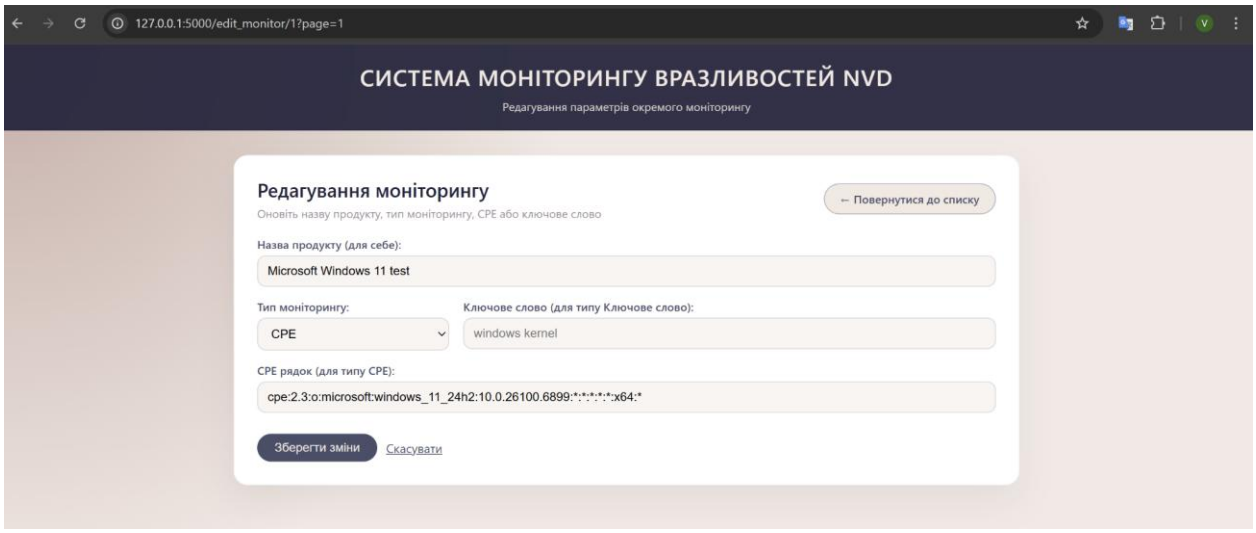

Рисунок 3.4 – Сторінка редагування параметрів моніторингу

Паралельно з розробкою веб-інтерфейсу було впроваджено сервісні функції, які відповідають за запуск повного циклу моніторингу. У окремому модулі було описано функцію, що отримує всі моніторинги з бази, проходить кожен з них, викликає модуль отримання даних з NVD, модуль фільтрації за CPE та ключовими словами, а потім модуль сповіщення для розсилки повідомлень. На етапі впровадження ця функція викликалася або з кнопки ручного запуску в інтерфейсі, або безпосередньо з командного рядка. Такий підхід дозволив послідовно перевірити коректність роботи кожного етапу, а також переконатися у стабільності всієї системи при послідовному запуску декількох моніторингів.

Заключним етапом впровадження стала організація автоматичного щоденного запуску перевірок без участі користувача. У середовищі Windows для цього було використано планувальник завдань Task Scheduler, який дає змогу запускати сценарії Python за розкладом [31]. Було створено нове завдання із

зрозумілою назвою, що описує його призначення, наприклад завдання щоденної перевірки NVD. Як тригер обрано щоденний запуск о 8 годині ранку, що дозволяє фахівцю з безпеки отримувати звіти на початку робочого дня. Як дію було налаштовано запуск програми, де в якості виконуваного файлу вказано інтерпретатор Python з каталогу віртуального середовища, а у параметрах командного рядка – шлях до сценарію, який здійснює запуск усіх моніторингів. Додатково задавався робочий каталог, щоб сценарій міг коректно знайти конфігураційні файли та базу даних. На рисунку 3.5 може бути наведений скріншот вікна Task Scheduler з налаштованим завданням щоденного запуску.

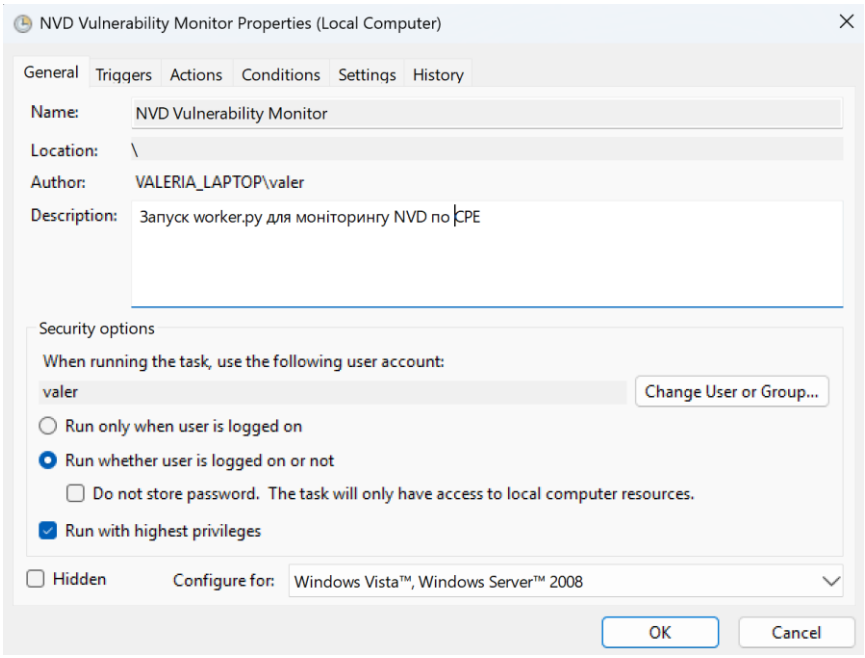

Рисунок 3.5 – Налаштування щоденного запуску скрипта моніторингу у Windows Task Scheduler

У результаті виконання всіх описаних кроків система була повністю впроваджена у робоче середовище: Flask-застосунок забезпечує веб-інтерфейс для керування моніторингами та отримувачами, база даних SQLite зберігає всі налаштування, а планувальник завдань гарантує регулярний автоматичний запуск перевірок у визначений час. Користувач отримує можливість гнучко змінювати конфігурації, додавати нові моніторинги, коригувати списки CPE та отримувачів,

при цьому не втручаючись у внутрішню логіку роботи модулів. Така послідовність впровадження демонструє, що розроблена система може бути інтегрована в реальну інфраструктуру та забезпечувати безперервний моніторинг вразливостей у повністю автоматизованому режимі.

3.2 Тестування системи

Тестування розробленої системи моніторингу вразливостей було спрямоване на перевірку того, наскільки коректно реалізовані функціональні вимоги, чи надійно працює збереження даних у базі, та чи стабільно виконується повний цикл моніторингу – від налаштування параметрів до надсилання електронних сповіщень. Основну увагу зосереджено на функціональному тестуванні з позиції користувача, тобто на перевірці сценаріїв використання системи без аналізу внутрішньої реалізації коду, що відповідає підходу тестування «чорної скриньки» [32, 33]. Такий підхід дає змогу оцінити, чи відповідає поведінка системи визначеним вимогам, і чи є вона достатньо зручною та передбачуваною для реального використання.

Першим груповим сценарієм було тестування процесу додавання основного отримувача листів, налаштування інтервалу перевірки в годинах та порогу важливості вразливостей. У веб-інтерфейсі відкривалася сторінка налаштувань, де заповнювалися поля з основною адресою електронної пошти, значенням інтервалу перевірки та мінімальним рівнем критичності (наприклад, за шкалою CVSS), з якого вразливості вважаються релевантними для сповіщення. Після натискання кнопки збереження система виконувала запис введених значень у базу даних SQLite. Далі проводилася перевірка коректності збереження: за допомогою інструменту перегляду SQLite відкривалась відповідна таблиця налаштувань, де перевірялося, що вказана електронна адреса, інтервал і поріг важливості збережені саме в тому вигляді, в якому були введені у формі. Після перезавантаження сторінки налаштувань значення полів автоматично підставлялися з бази даних, що підтверджувало замкнутий цикл «ввід–збереження–відображення». На рисунку 3.6

доцільно навести скріншот форми з заповненими полями, а на рисунку 3.7 – фрагмент таблиці бази даних з відповідним записом.

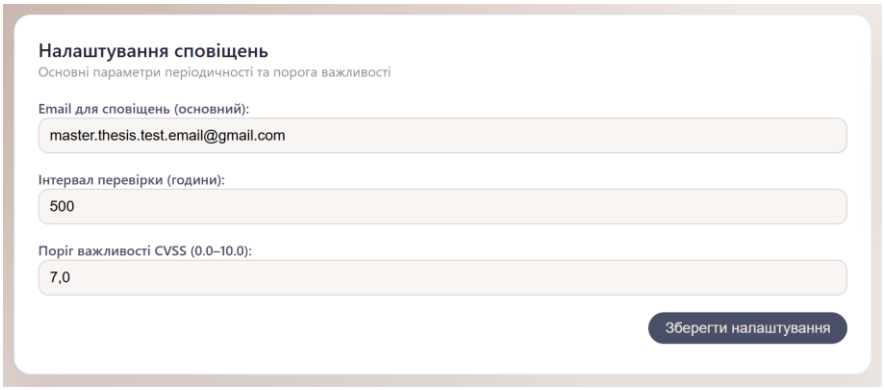

Рисунок 3.6 – Форма налаштування основного отримувача, інтервалу перевірки та порогу важливості

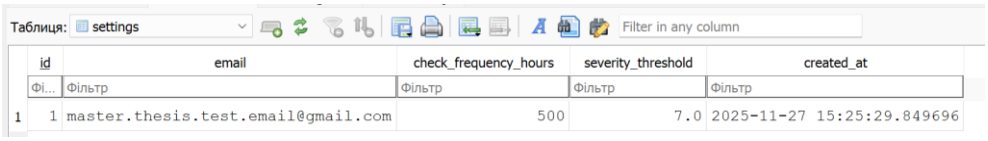

Рисунок 3.7 – Фрагмент таблиці бази даних з параметрами основного отримувача та інтервалу перевірки

Наступним етапом було тестування роботи з допоміжними отримувачами. Спочатку перевірявся стан, коли додаткові отримувачі ще не налаштовані: сторінка відповідного розділу відображала повідомлення про відсутність записів або порожній список. Це дозволило підтвердити коректну обробку ситуації «порожніх даних», яка є важливою складовою стабільності веб-застосунків [32]. Далі через форму додавання вводилися одна або кілька додаткових електронних адрес, після чого система здійснювала запис нових записів у таблицю додаткових отримувачів, пов’язуючи їх з конкретним моніторингом. Після збереження сторінка оновлювалася, і у веб-інтерфейсі з’являвся список додаткових отримувачів з можливістю їх подальшого видалення. Аналогічно, через SQLite-переглядач перевірялося, що в базі даних з’явилися нові рядки з коректними адресами та прив’язкою до потрібного моніторингу. На рисунку 3.8 доцільно показати

початковий порожній список додаткових отримувачів, на рисунку 3.9 – сторінку після їх додавання, а на рисунку 3.10 – відповідний стан таблиці у базі даних.

Рисунок 3.8 – Сторінка додаткових отримувачів до додавання записів

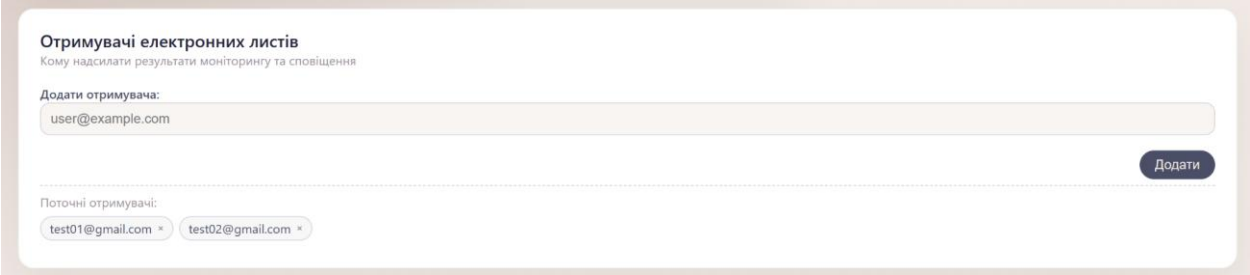

Рисунок 3.9 – Сторінка додаткових отримувачів після додавання користувачів

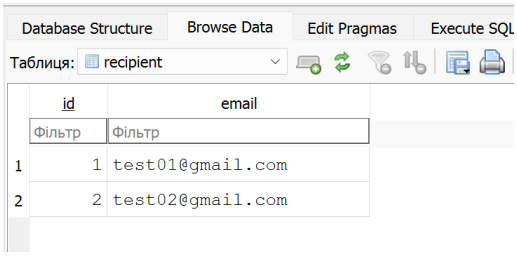

Рисунок 3.10 – Фрагмент таблиці бази даних з додатковими отримувачами

Окремий набір тестів був присвячений роботі зі списком моніторингів. Спершу перевірялося створення моніторингу за CPE. У формі створення моніторингу задавалася назва, тип моніторингу «за CPE», один або кілька CPE-ідентифікаторів у відповідному полі, а також інтервал перевірки. Після збереження моніторинг з’являвся у загальному списку на головній сторінці, де було видно його назву, тип, інтервал та інші параметри. Далі у базі даних перевірялося, що для нового запису коректно збережено тип моніторингу, текстове поле зі списком CPE та пов’язані параметри. Аналогічний сценарій виконували для моніторингу за

ключовим словом: змінювався тип моніторингу, заповнювалося поле ключових слів, а поле CPE залишалося порожнім. Після збереження у списку моніторингів можна було візуально відрізнити записи за типом, а в базі даних – переконатися, що правильно використані відповідні поля. На рисунках 3.11 і 3.12 можна розмістити скріншоти форм створення моніторингу за CPE та за ключовими словами, а на рисунку 3.13 – вигляд списку моніторингів із обома типами записів.

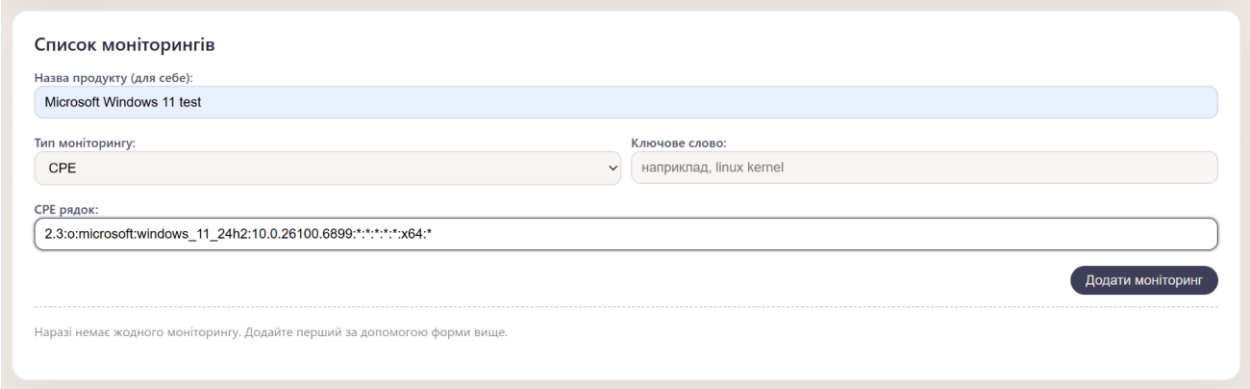

Рисунок 3.11 – Створення моніторингу за CPE

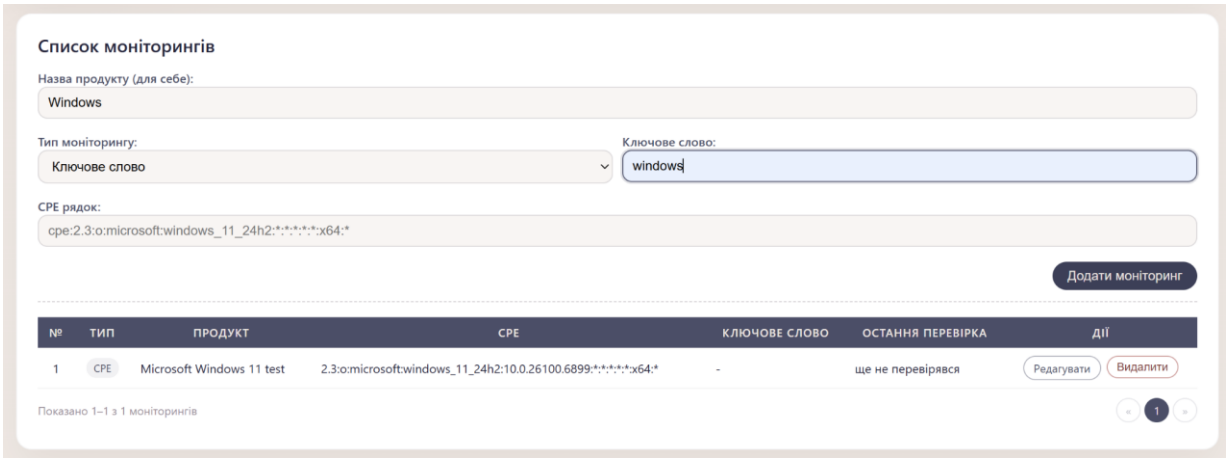

Рисунок 3.12 – Створення моніторингу за ключовим словом

Рисунок 3.13 – Список моніторингів з моніторингами за CPE та за ключовими словами

Далі тестувався процес редагування і видалення моніторингів. Для редагування обирався один із наявних моніторингів, відкривалася сторінка редагування, де автоматично підставлялися поточні параметри. У формі змінювалися окремі поля, наприклад, назва моніторингу, інтервал перевірки або список CPE, після чого дані знову зберігалися. Перевірка полягала в тому, що у списку моніторингів відображалися оновлені значення, а у відповідному рядку бази даних змінювалися саме ті поля, які було відредаговано. Сценарій видалення полягав у натисканні кнопки видалення для вибраного моніторингу, підтвердженні операції та перевірці того, що запис зникає як зі списку в інтерфейсі, так і з таблиці в базі даних. Такий підхід відповідає рекомендаціям щодо перевірки CRUD-операцій (create, read, update, delete) у веб-застосунках [33]. На рисунку 3.14 доцільно показати сторінку редагування моніторингів, на рисунку 3.15 – список моніторингів після редагування, а на рисунку 3.16 – стан списку після видалення одного з моніторингів.

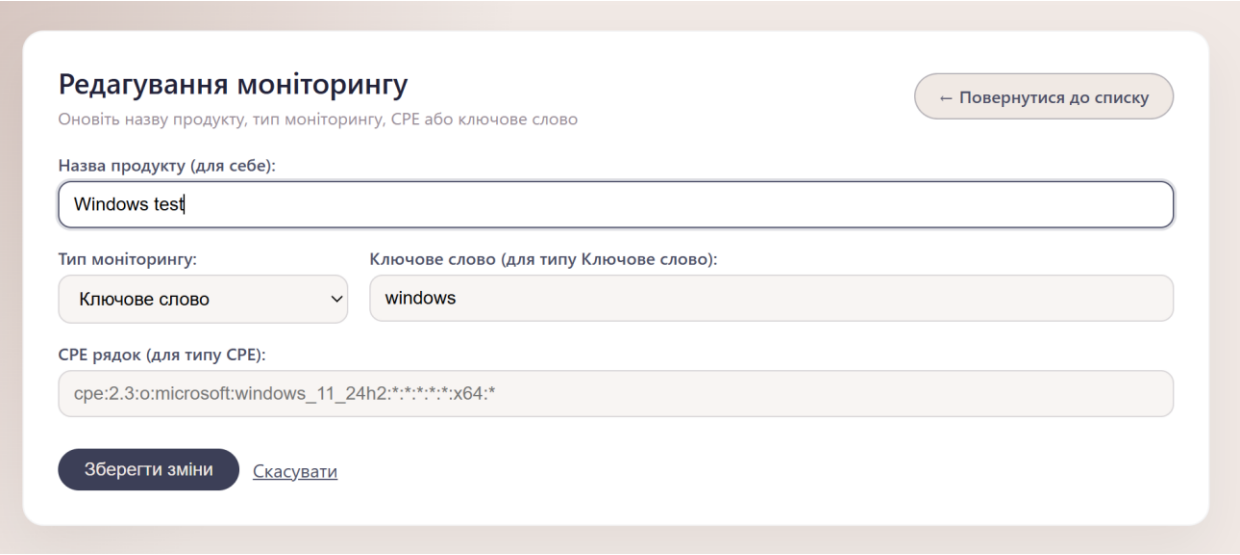

Рисунок 3.14 – Сторінка редагування моніторингу

Рисунок 3.15 – Список моніторингів після редагування параметрів

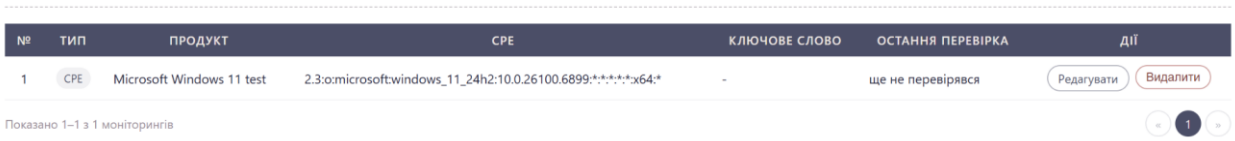

Рисунок 3.16 – Список моніторингів після видалення запису

Наступний блок тестів був пов’язаний із перевіркою коректності виконання самих перевірок вразливостей та формування електронних листів. Спершу за допомогою кнопки ручного запуску на сторінці налаштувань ініціювався повний цикл моніторингу. Перед запуском фіксували поточний стан таблиці моніторингів, зокрема дату та час останньої перевірки для кожного запису. Після завершення ручного запуску сторінка оновлювалася, і значення дати та часу останньої перевірки змінювалися відповідно до фактичного моменту виконання. Це підтверджувало, що система не лише звернулася до NVD і опрацювала дані, а й коректно оновила метаінформацію про перевірки. На рисунку 3.17 може бути показаний вигляд таблиці моніторингів після ручного запуску, де видно оновлені поля дати та часу.

Рисунок 3.17 – Оновлені значення дати та часу останньої перевірки після ручного запуску

Паралельно перевірялася робота модуля сповіщення. Для моніторингів, налаштованих на продукти з відомими вразливостями, після ручного запуску моніторингу перевірялася поштова скринька основного отримувача, а також, при необхідності, скриньки додаткових отримувачів. У вхідних повідомленнях шукали листи від налаштованої облікової записи Gmail, яка використовується системою для надсилання сповіщень. Перевірялися тема листа (наявність назви моніторингу

або іншого ідентифікатора), коректність тексту повідомлення, список знайдених вразливостей, а також відсутність дублювання листів при повторних запусках без появи нових релевантних записів. Таким чином, було підтверджено, що модуль сповіщень формує зміст листа відповідно до результатів фільтрації, а механізм доставки через SMTP працює стабільно. На рисунку 3.18 доцільно показати загальний вигляд списку вхідних листів із системи, а на рисунку 3.19 – вміст одного з листів із переліком виявлених вразливостей.

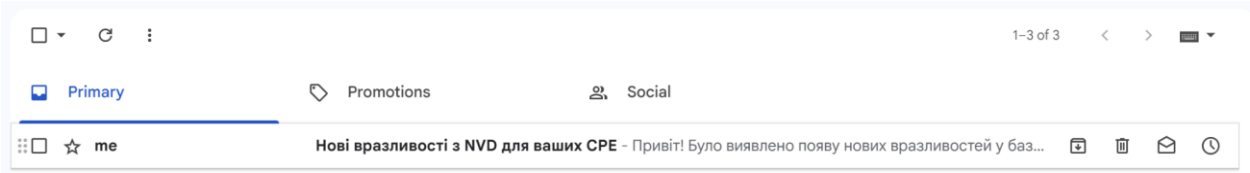

Рисунок 3.18 – Список вхідних листів, отриманих від системи моніторингу

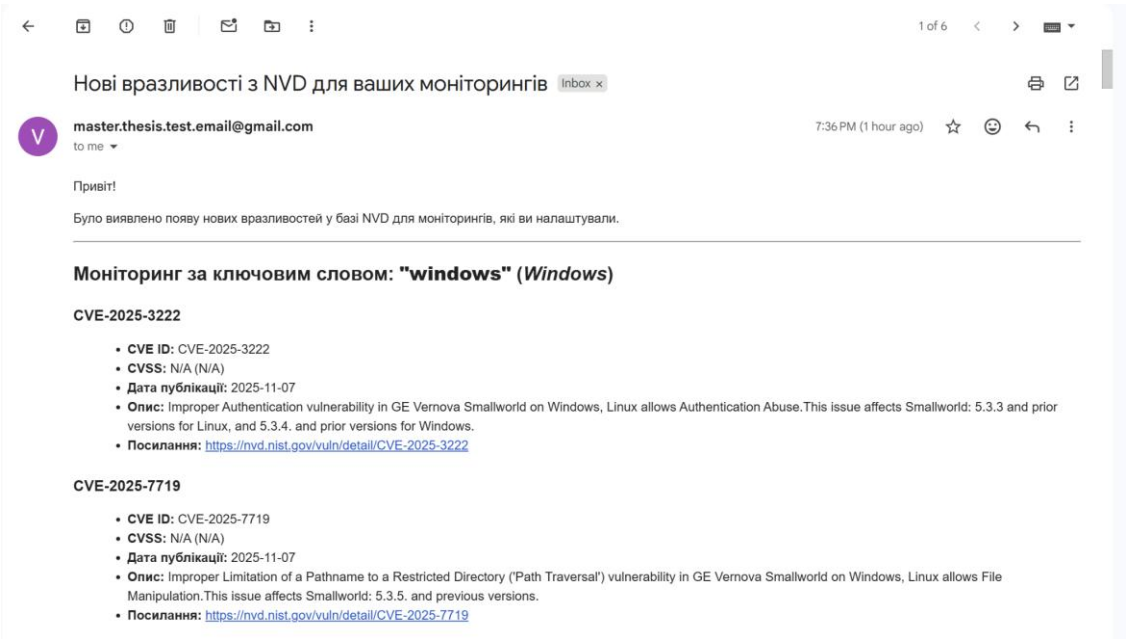

Рисунок 3.19 – Вигляд листа зі списком знайдених вразливостей

Завершальним етапом тестування було підтвердження коректності роботи автоматичної перевірки, яка виконується щоденно за розкладом. Після налаштування завдання у Windows Task Scheduler тестування проводилося у два кроки. Спочатку запуск системи очікувався у заданий час, наприклад о 8:00. Після цього у веб-інтерфейсі відкривалася сторінка зі списком моніторингів, де

перевірялося, що поле дати та часу останньої перевірки для всіх активних моніторингів оновлено саме на цей момент часу. Далі перевірялися поштові скриньки основного та додаткових отримувачів: очікувалося, що після автоматичного запуску були надіслані листи з актуальною інформацією про знайдені вразливості, або ж, за відсутності нових записів, листи не надсилалися. Такий підхід відповідає рекомендаціям стандартів щодо документування та відслідковування результатів тестування, коли кожен запуск тесту повинен мати чітко фіксований час виконання та спостережувані результати [34]. На рисунку 3.20 можна показати список моніторингів після виконання автоматичної перевірки, а на рисунку 3.21 – нові листи у поштовій скриньці, отримані саме в час, який відповідає розкладу.

Рисунок 3.20 – Оновлений час останньої перевірки після автоматичного запуску за розкладом

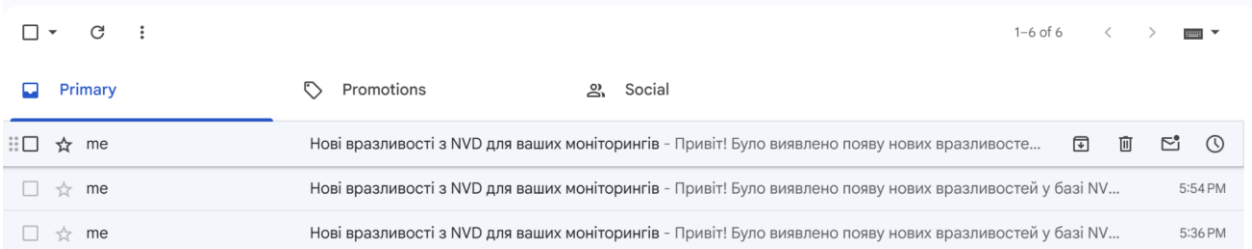

Рисунок 3.21 – Вигляд поштової скриньки після автоматичної ранкової перевірки

У результаті проведених тестових сценаріїв було підтверджено, що система коректно зберігає всі налаштування в базі даних, забезпечує узгодженість між веб-інтерфейсом і фактичними даними SQLite, правильно обробляє сценарії створення, редагування та видалення моніторингів, а також стабільно виконує ручні та автоматичні перевірки з подальшим надсиланням електронних сповіщень. Це

свідчить про відповідність реалізованої системи її функціональним вимогам і готовність до використання в реальному середовищі експлуатації.

3.3 Аналіз отриманих результатів та оцінка ефективності

Результати тестування показали, що розроблена система в цілому відповідає початково сформульованій меті: забезпечити автоматизований моніторинг вразливостей з бази NVD з можливістю фільтрації за CPE та ключовими словами і розсилкою сповіщень електронною поштою. У всіх перевірених сценаріях користувач мав змогу самостійно налаштувати основного та додаткових отримувачів, створити необхідні моніторинги для конкретних продуктів, запустити перевірку вручну чи очікувати її автоматичного виконання за розкладом та отримати узгоджені за змістом звіти на електронну пошту. Це означає, що ключові функціональні вимоги, сформульовані у розділах аналізу та проектування, реалізовано і підтверджено на практиці.

Аналіз роботи системи в ході тестування показав, що зв’язок між веб інтерфейсом та базою даних є коректним і послідовним. Усі операції створення, редагування та видалення моніторингів, а також додавання допоміжних отримувачів відображалися як у графічному інтерфейсі, так і в таблицях SQLite без розсинхронізації. Зокрема, при зміні типу моніторингу або списку CPE коректно оновлювалися відповідні поля у базі, а при повторному відкритті форм у веб інтерфейсі користувач бачив актуальні значення. Це свідчить про відсутність критичних помилок у логіці збереження та завантаження даних та підтверджує цілісність внутрішньої моделі налаштувань.

Ще одним важливим аспектом аналізу є перевірка повного циклу моніторингу. У тестових сценаріях для кількох моніторингів були обрані продукти, для яких у NVD гарантовано є актуальні записи. Після ручного запуску система зверталася до API NVD, отримувала перелік вразливостей, відфільтровувала їх за CPE та ключовими словами, а потім формувала електронний лист. У перевірених випадках у листах містилися саме ті CVE, які очікувалися для вибраних продуктів,

без сторонніх записів. Тобто логіка фільтрації працювала коректно, а модуль сповіщення коректно відображав результати фільтрації у тексті повідомлень. Відсутність дублювання листів при повторних запусках без появи нових релевантних вразливостей показала, що система не створює зайвого інформаційного шуму і не перевантажує отримувачів однаковими сповіщеннями.

Окремо було оцінено часові показники роботи системи. У тестовому середовищі, де одночасно налаштовано кілька моніторингів, повний цикл перевірки, що включає звернення до NVD, обробку відповіді, фільтрацію та розсилку листів, тривав у межах кількох секунд для середньої кількості знайдених вразливостей. Навіть з урахуванням додаткового часу на встановлення TLS з'єднання з поштовим сервером та формування листів загальна тривалість виконання не перевищувала десятків секунд. Це означає, що щоденний автоматичний запуск у заданий час практично не створює відчутного навантаження на систему і дозволяє отримувати результати моніторингу майже відразу після старту завдання. Для користувача це проявляється у тому, що актуальні звіти опиняються в поштовій скриньці вже на початку робочого дня без помітних затримок.

Якщо порівнювати запропоноване рішення з альтернативними підходами, які були розглянуті раніше, можна виділити кілька суттєвих переваг. У порівнянні з ручним пошуком у веб інтерфейсі NVD система усуває потребу щоденно заходити на сайт, задавати фільтри та переглядати списки вразливостей, а також самостійно відбирати лише ті записи, які стосуються конкретних продуктів. Усе це виконується автоматично відповідно до заздалегідь налаштованих моніторингів. У порівнянні з важкими корпоративними рішеннями для управління вразливостями перевагою є простота розгортання, відсутність потреби у встановленні агентів на хости, прозора логіка роботи та можливість гнучко підлаштувати опис моніторингів під конкретні потреби невеликої команди або окремого спеціаліста. Система не претендує на заміну повномасштабних сканерів інфраструктури, але ефективно закриває нішу точкового моніторингу продуктів за даними NVD.

Важливою складовою оцінки ефективності є також аналіз того, наскільки система допомагає зменшити часові витрати фахівця з безпеки. За рахунок автоматизації перевірок та доставки результатів на електронну пошту відпадає необхідність виконувати рутинні повторювані дії. Користувач отримує стислий структурований звіт і може одразу переходити до прийняття рішень щодо реакції на виявлені вразливості, замість витрачати час на формування цього звіту вручну. Навіть при невеликій кількості моніторингів щоденний виграш у часі для спеціаліста є відчутним, а при збільшенні кількості продуктів, що контролюються, вигода від автоматизації стає ще більш очевидною.

Разом з тим аналіз результатів показав і певні обмеження рішення. Система залежить від актуальності та повноти даних NVD і не виконує активного сканування реальної інфраструктури, не перевіряє фактичні версії встановленого програмного забезпечення та стан його оновлення. Вона також використовує електронну пошту як основний канал сповіщення, що є зручним, але не завжди достатнім у випадку великих організацій, які можуть потребувати інтеграції з квитковими системами, SIEM чи іншими платформами. Однак з архітектурного погляду система спроєктована так, щоб у майбутньому можна було розширити набір каналів сповіщення, додати кешування та збереження історії виявлених вразливостей, а також інтегруватися з іншими інструментами.

Узагальнюючи, можна зробити висновок, що за результатами тестування та аналізу роботи розроблена система досягає поставленої мети та демонструє достатню ефективність у своїй цільовій області застосування. Вона забезпечує автоматизований моніторинг вразливостей з фільтрацією за CPE та ключовими словами, надає гнучкий механізм керування моніторингами та списками отримувачів, а також гарантує своєчасне сповіщення про знайдені загрози. З урахуванням виявлених обмежень система може розглядатися як легкий, але дієвий інструмент для підсилення процесу управління вразливостями, який особливо доречний у середовищах, де повноцінні комплексні платформи є надмірними або економічно недоцільними.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Експлуатація веборієнтованої системи моніторингу вразливостей на основі NVD та ідентифікаторів CPE пов’язана не лише з вимогами кібербезпеки, а й із необхідністю забезпечення безпечних умов праці користувачів комп’ютерної мережі, які щоденно працюють із цією системою. Тривала робота за ПК, висока концентрація уваги під час аналізу технічної інформації та постійний контакт з мережевим обладнанням формують комплекс факторів ризику для здоров’я працівників, що прямо підкреслюється у профільних методичних посібниках з безпеки в надзвичайних ситуаціях. Тому в межах даного розділу розглядаються організаційні та технічні заходи охорони праці, а також фактори ризику і можливі порушення здоров’я користувачів комп’ютерної мережі з урахуванням вимог чинного законодавства України та рекомендацій фахової літератури.

4.1 Охорона праці

Під час розробки системи моніторингу вразливостей та підготовки до її впровадження враховувались вимоги охорони праці і пожежної безпеки, зокрема на рівні організації робочого процесу розробника, налаштування робочого місця з ПК, а також вимоги до безпечної експлуатації комп’ютерного та мережевого обладнання, яке забезпечує роботу вебзастосунку і планових перевірок.

Специфіка виконання робіт у межах розробки системи моніторингу вразливостей визначається тим, що розроблення, тестування і супровід системи відбуваються переважно за комп’ютером, а експлуатація системи передбачає наявність постійно доступного робочого середовища, мережевого підключення та електроживлення. Відповідно до загальних вимог охорони праці пріоритет надавався профілактиці ризиків, що виникають під час тривалої статичної пози, напруження зору, психоемоційного навантаження під час налагодження і тестування, а також ризиків, пов’язаних із використанням електрообладнання та можливими пожежонебезпечними ситуаціями [35]. Під час розробки організаційно

було орієнтовано процес на скорочення непотрібних ручних операцій під час експлуатації. Зокрема, для регулярних перевірок уразливостей було реалізовано механізм автоматичного запуску (планова перевірка) та можливість ручного запуску з інтерфейсу. Такий підхід зменшує потребу у тривалому безперервному спостереженні за процесом, а отже опосередковано знижує тривалість напруженої роботи користувача за ПК і ризики перевтоми.

Важливим елементом є безпечна організація робочого місця користувача персональним комп'ютером. Вимоги до роботи з екранними пристроями передбачають, що робоче місце має бути організоване так, аби мінімізувати вплив відблисків, забезпечити зручну позу, можливість змінювати положення тіла, а також зменшити зорове напруження за рахунок коректного розташування екрана, клавіатури та вхідних пристроїв [36]. Під час виконання робіт у проєкті це враховувалось через застосування стандартного робочого середовища розробника, використання монітора з адекватною яскравістю і контрастністю, налаштування масштабування інтерфейсу редактора коду, а також через обмеження тривалих безперервних сесій за рахунок планування робіт етапами (розроблення, тестування, документування) з перервами відповідно до самопочуття. Окремо було враховано, що частина операцій системи має виконуватися у фоновому режимі, а взаємодія користувача з системою зосереджується на коротких адміністративних діях, налаштуванні моніторингів та перегляді результатів.

Санітарні вимоги до мікроклімату є практично важливими для приміщень, де виконуються роботи з використанням ПК, оскільки відхилення температури, вологості та швидкості руху повітря впливають на працездатність, стомлюваність і загальне самопочуття [37]. Для робіт, характерних для розробника та адміністратора інформаційної системи, типовою є категорія легких робіт, для яких у нормах наведені оптимальні параметри мікроклімату. У межах підготовки робочого середовища для розроблення і тестування було забезпечено провітрювання, підтримання комфортної температури та уникнення перегріву обладнання.

Оскільки система є веборієнтованою і передбачає роботу з комп'ютерною технікою та мережевими пристроями, окремо враховувались вимоги електробезпеки. Електробезпека у даному випадку включає безпечне підключення комп'ютера, маршрутизатора, мережевих адаптерів та інших елементів робочого місця до електромережі, недопущення використання пошкоджених кабелів і подовжувачів, а також запобігання перевантаженню електричних ліній, що може призвести до ураження електричним струмом або створення пожежонебезпечної ситуації. Під час експлуатації системи було обрано підхід, за якого критично важливі компоненти можуть працювати у стабільному режимі без необхідності частих фізичних втручань у підключення. Для цього сценарії запуску автоматичної перевірки реалізується засобами планувальника завдань операційної системи, що усуває потребу залишати відкритими зайві інструменти та зменшує кількість ручних маніпуляцій із обладнанням. У практичних рекомендаціях для користувача системи окремо фіксується необхідність підключення робочого місця до справної електромережі з наявним захисним заземленням, використання сертифікованих мережевих фільтрів і уникнення роботи з обладнанням у разі виявлення нагрівання вилок, розеток або блоків живлення.

Пожежна безпека для робочих приміщень, де розміщується комп'ютерне обладнання, є критичною через наявність електромережі, джерел живлення, акумуляторів, а також постійне тепловиділення від пристроїв. У правилах пожежної безпеки визначено загальні вимоги до утримання приміщень, шляхів евакуації, експлуатації електрообладнання, а також недопущення використання несправних електроприладів чи саморобних подовжувачів [38]. На рівні організації експлуатації системи моніторингу це враховується тим, що робоче місце, з якого здійснюється адміністрування системи, має бути забезпечене вільним доступом до вимикачів живлення, а розміщення обладнання повинно виключати перекриття вентиляційних отворів, накопичення пилу в місцях активного теплообміну та контакт із легкозаймистими матеріалами. Умови цивільного захисту підкреслюють необхідність запобігання надзвичайним ситуаціям та готовності до реагування, що на практиці означає наявність базових інструкцій дій у разі задимлення, короткого

замикання, пожежі, а також підтримання працездатних засобів первинного пожежогасіння у приміщенні, де розташоване обладнання [39].

З огляду на те, що система моніторингу вразливостей використовується у складі інформаційної інфраструктури організації або робочої групи, важливими є також організаційні заходи охорони праці. Вони охоплюють інструктажі, визначення відповідальності за безпечну експлуатацію робочого місця, дотримання режиму праці та відпочинку, контроль технічного стану обладнання та кабельного господарства, а також фіксацію правил безпечної роботи з ПК і електрообладнанням [35]. Для даного проєкту такі заходи відображаються у супровідній документації, де описується рекомендований порядок розгортання, регулярні перевірки працездатності системи, безпечні умови підключення та перелік дій у разі аварійних ситуацій. Додатково, з урахуванням специфіки системи, яка виконує запити до зовнішнього сервісу та надсилає повідомлення електронною поштою, у документації доцільно підкреслити, що автоматизація процесів має виконуватись із мінімізацією непередбачуваних ручних втручань, оскільки це знижує ризик людських помилок і непродуктивного навантаження на користувача, який працює за ПК.

Таким чином, під час розробки системи моніторингу вразливостей вимоги охорони праці були враховані як на рівні організації робочого місця та режиму роботи розробника, так і на рівні рекомендацій з експлуатації системи. Практична реалізація веборієнтованої системи з автоматизованими перевірками і сповіщеннями сприяє скороченню тривалих ручних операцій, а врахування вимог електро та пожежної безпеки зменшує ризики, пов’язані з роботою комп’ютерного та мережевого обладнання [38].

4.2 Фактори ризику і можливі порушення здоров’я користувачів комп’ютерної мережі

Користувачі комп’ютерної мережі, які працюють із системою моніторингу вразливостей на основі NVD та ідентифікаторів CPE, щоденно взаємодіють з

комп’ютерною технікою, мережевим обладнанням та інтерфейсами, що відображають значні обсяги технічної інформації. Такий характер діяльності поєднує вплив фізичних, психофізіологічних та організаційних факторів ризику, які за відсутності профілактичних заходів можуть призвести до порушень здоров’я працівників і зниження їхньої працездатності [40, 41].

Одним із базових факторів ризику є тривала статична робоча поза при роботі за персональним комп’ютером. У процесі налаштування моніторингів, аналізу звітів, перегляду виявлених записів з NVD та формування рішень користувачі довгий час залишаються у сидячому положенні з незначною амплітудою рухів. Якщо висота робочого столу та стільця, розташування монітора, клавіатури й миші не відповідають ергономічним вимогам, відбувається хронічне перенапруження м’язів шийно-плечового відділу та спини, порушується кровообіг, підвищується ризик формування остеохондрозу, болю в попереку та суглобах верхніх кінцівок. Це особливо небезпечно при тривалих робочих змінах, характерних для фахівців з кібербезпеки.

Додатковим значущим фактором є зорове навантаження. Інтерфейс системи моніторингу вразливостей містить таблиці, списки, текстові описи, ідентифікатори CPE та параметри фільтрації, що потребують постійної зорової концентрації. Робота з дрібним шрифтом, часті перемикання між вікнами, аналіз логів та технічної документації, а також некоректно налаштовані параметри яскравості й контрастності монітора спричинюють втому очей, відчуття сухості та печіння, головний біль наприкінці робочого дня. Нормативні та методичні матеріали рекомендують забезпечувати раціональну організацію освітлення, оптимальну відстань до екрана, належні кути огляду та регулярні перерви для відпочинку зору [40, 41].

Важливу роль відіграють мікрокліматичні умови та стан повітряного середовища у приміщеннях, де розміщуються робочі місця користувачів і мережеве обладнання. Сервери, маршрутизатори, комутатори, джерела безперебійного живлення та персональні комп’ютери виділяють значну кількість тепла, що може спричинити перегрів повітря, його пересушування та зниження вмісту кисню у разі

недостатньої вентиляції. Такі умови призводять до загальної втоми, сонливості, погіршення концентрації уваги, дратівливості та зниження якості прийняття рішень у процесі аналізу результатів моніторингу. Дотримання нормативних параметрів температури, вологості та швидкості руху повітря є важливою складовою забезпечення безпечної роботи користувачів комп'ютерної мережі.

До фізичних чинників також належить шум від роботи комп'ютерного обладнання, систем охолодження, вентиляційних установок та серверних шаф. Постійний фоновий шум, навіть якщо його рівень не перевищує гранично допустимі значення, за тривалої дії сприяє підвищенню стомлюваності, погіршенню концентрації уваги, збільшенню кількості помилок під час виконання рутинних операцій, таких як перевірка параметрів моніторингу, аналіз критичності вразливостей чи підготовка звітів. У середовищі, де обробляються дані про безпеку, це може опосередковано впливати і на загальний рівень інформаційної безпеки організації.

Значну роль відіграють психофізіологічні фактори, насамперед стрес та інформаційне перевантаження. Робота із системою моніторингу вразливостей передбачає постійне відстеження нових загроз, аналіз CVE-записів, ухвалення рішень щодо пріоритизації оновлень та інформування відповідальних осіб. У періоди інтенсивного оновлення бази NVD або підвищеної активності кіберзагроз навантаження на користувачів системи суттєво зростає, що може викликати емоційне виснаження, тривожність, розлади сну та зниження стійкості до стресу. Методичні матеріали з безпеки в надзвичайних ситуаціях і цивільної безпеки наголошують на тому, що такі психоемоційні навантаження, за відсутності профілактичних заходів, можуть у подальшому переходити в хронічні стани [40, 41].

Окрема група факторів ризику має організаційний характер. Відсутність систематичних інструктажів з охорони праці, недостатня кількість навчань щодо дій у надзвичайних ситуаціях, нечітке розподілення обов'язків при виникненні аварій, пожеж або загроз техногенного характеру можуть призвести до неузгоджених дій персоналу. Працівники, які обслуговують систему моніторингу

вразливостей і розміщене поруч мережеве обладнання, повинні чітко знати маршрути евакуації, місця розташування первинних засобів пожежогасіння, пункти збору та порядок інформування відповідальних осіб. Недотримання цих вимог збільшує ймовірність травм, дезорганізації роботи та посилення негативних наслідків надзвичайних ситуацій.

У реальних умовах зазначені фактори ризику рідко проявляються ізольовано. Як правило, вони діють комплексно: незручна робоча поза поєднується з високим рівнем зорового та інформаційного навантаження, несприятливим мікрокліматом, шумом та організаційними недоліками. Для працівників, які підтримують працездатність системи моніторингу вразливостей і приймають рішення щодо реагування на загрози, така сукупність факторів може призводити до помилок, зниження швидкості реакції, довготривалих функціональних порушень здоров'я. Саме тому методичні посібники з безпеки в надзвичайних ситуаціях та техноекології рекомендують оцінювати ризики комплексно, враховуючи як умови звичайної роботи, так і можливість розвитку надзвичайних ситуацій [40, 41].

Отже, діяльність користувачів комп'ютерної мережі, які працюють із системою моніторингу вразливостей, супроводжується впливом фізичних, зорових, мікрокліматичних, психофізіологічних та організаційних факторів ризику. Без своєчасного впровадження профілактичних заходів це може призвести до функціональних порушень здоров'я, зниження працездатності та ефективності прийняття рішень у сфері кібербезпеки. Урахування вимог охорони праці та рекомендацій профільних методичних посібників дозволяє мінімізувати вплив цих факторів і забезпечити безпечну, стійку до надзвичайних ситуацій експлуатацію розробленої системи.

ВИСНОВКИ

У кваліфікаційній роботі розв’язано науково технічну задачу розробки системи моніторингу вразливостей на основі даних National Vulnerability Database із використанням фільтрації за Common Platform Enumeration та автоматичними сповіщеннями електронною поштою. Побудовано цілісний підхід, який поєднує формалізований опис програмних продуктів, автоматизоване отримання даних про вразливості та доставку релевантної інформації відповідальним фахівцям без необхідності щоденного ручного перегляду публічних баз.

У теоретичній частині роботи проаналізовано предметну область моніторингу вразливостей, розглянуто структуру та можливості NVD, а також стандарти ідентифікації вразливостей та продуктів, зокрема CVE та CPE. Показано, що використання CPE як формального опису програмних продуктів дає змогу суттєво підвищити точність відбору записів з NVD порівняно з неконтрольованим пошуком за довільними текстовими запитами. Сформульовано вимоги до програмної системи, розроблено архітектуру, структуру бази даних та модель основних програмних компонентів.

Практичним результатом роботи є реалізація веборієнтованої системи, що дозволяє налаштовувати окремі профілі моніторингу для різних груп програмних продуктів, задавати список CPE та ключових слів, порогові значення критичності вразливостей і списки отримувачів сповіщень. Система автоматично звертається до NVD, виконує фільтрацію отриманих даних і формує структуровані електронні листи з переліком актуальних вразливостей, які надсилаються основному та додатковим отримувачам. Реалізовано як ручний запуск перевірки через веб інтерфейс, так і повністю автоматичний запуск за розкладом операційної системи.

Наукова новизна роботи полягає у поєднанні використання CPE для формалізованого опису програмних продуктів із механізмом конфігурованих профілів моніторингу, які можуть незалежно налаштовуватися для різних підсистем, а також у застосуванні комбінованої фільтрації за CPE та ключовими словами з урахуванням порогу критичності. Запропонований підхід дозволяє не

лише автоматизувати отримання даних з NVD, а й отримувати стислий, орієнтований на конкретні потреби звіт, який безпосередньо підтримує прийняття рішень щодо реагування на вразливості.

Під час тестування системи у типових сценаріях використання підтверджено коректність роботи основних модулів, цілісність збереження даних у базі SQLite та узгодженість між веб інтерфейсом, внутрішньою логікою та реально отриманими повідомленнями електронної пошти. Для кількох налаштованих моніторингів повний цикл перевірки від моменту запуску до формування листів займає від кількох до кількох десятків секунд, що дає змогу оперативно отримувати актуальну інформацію на початку робочого дня. Достовірність отриманих результатів підтверджується відтворюваністю тестових сценаріїв, використанням офіційних джерел даних та відповідністю виявлених вразливостей очікуваним записам у NVD.

Розроблена система може бути рекомендована до використання у невеликих організаціях, навчальних лабораторіях та індивідуальній практиці фахівців з інформаційної безпеки як допоміжний інструмент у процесі управління вразливостями. Вона дає змогу зменшити обсяг рутинних операцій, підвищити своєчасність отримання інформації про загрози та структуровано організувати процес моніторингу. Перспективами подальшого розвитку є інтеграція додаткових джерел даних про вразливості, розширення каналів сповіщення, доповнення системи засобами аналітики та побудови історичних звітів, а також інтеграція з іншими компонентами інфраструктури кіберзахисту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. National Vulnerability Database (NVD). National Institute of Standards and Technology. URL: https://nvd.nist.gov/ (дата звернення: 20.09.2025).
2. Scarfone K., Mell P. The Common Configuration Scoring System (CCSS): Metrics for Software Security Configuration. Gaithersburg: NIST, 2007. 26 p. URL: https://csrc.nist.gov/publications/detail/nistir/7502/final (дата звернення: 20.09.2025).
3. National Institute of Standards and Technology. National Vulnerability Database (NVD). URL: https://nvd.nist.gov/ (дата звернення: 26.09.2025).
4. National Institute of Standards and Technology. Vulnerability JSON Data Feeds. URL: https://nvd.nist.gov/vuln/data-feeds (дата звернення: 27.09.2025).
5. Waltermire D., Booth H., Scarfone K. The Technical Specification for the CPE Naming Scheme. Version 2.3. NIST Interagency Report 7695, Revision 1. Gaithersburg: NIST, 2016. 38 p. URL: https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.7695r1.pdf (дата звернення: 01.10.2025).
6. Vulners. Vulnerability Intelligence Platform. URL: https://vulners.com/ (дата звернення: 01.10.2025).
7. Snyk. Developer-First Security. URL: https://snyk.io/ (дата звернення: 04.10.2025).
8. Sommerville I. Software Engineering. 10th ed. Boston: Pearson, 2016. 792 p.
9. Google. Sign in with App Passwords. URL: https://support.google.com/accounts/answer/185833?hl=en (дата звернення: 04.10.2025).
10. Hipp D. R. SQLite – Lightweight Database Engine. URL: https://sqlite.org/index.html (дата звернення: 10.10.2025).
11. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. 7th ed. New York: McGraw-Hill, 2020. 1376 p.

12. Date C. J. An Introduction to Database Systems. 8th ed. Boston: Pearson, 2003. 1024 p.

13. Петрик М. Р., Петрик О. Ю. Моделювання програмного забезпечення. Тернопіль: ТНТУ ім. І. Пулюя, 2015. 144 с.

14. Застосування UML (частина 3). Діаграма класів – Class Diagram // Державний університет телекомунікацій. 19.02.2020. URL: https://duikt.edu.ua/ua/news-1-626-8002-zastosuvannya-uml-chastina-3-diagrama-klasiv----class-diagram_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy (дата звернення: 14.10.2025).

15. UML діаграми, їхні основні типи та процес розроблення // FoxmindEd. 06.04.2024. URL: https://foxminded.ua/uml-diagramy/ (дата звернення: 15.10.2025).

16. What Is the National Vulnerability Database (NVD)? // Fortinet Cyber Glossary. URL: https://www.fortinet.com/resources/cyberglossary/national-vulnerability-database-nvd (дата звернення: 20.10.2025).

17. Vulnerability APIs – NVD // National Vulnerability Database. URL: https://nvd.nist.gov/developers/vulnerabilities (дата звернення: 21.10.2025).

18. Data Feeds – NVD // National Vulnerability Database. URL: https://nvd.nist.gov/vuln/data-feeds (дата звернення: 22.10.2025).

19. Reitz K. Requests Documentation. Release 2.32.5. URL: https://buildmedia.readthedocs.org/media/pdf/requests/latest/requests.pdf (дата звернення: 22.10.2025).

20. Python Software Foundation. JSON encoder and decoder. Python 3 Standard Library Documentation. URL: https://docs.python.org/uk/3/library/json.html (дата звернення: 02.11.2025).

21. Security Content Automation Protocol (SCAP). Common Platform Enumeration (CPE) // National Institute of Standards and Technology. URL: https://csrc.nist.gov/projects/security-content-automation-protocol/specifications/cpe (дата звернення: 02.11.2025).

22. Cheikes B. A., Waltermire D. et al. Common Platform Enumeration: Naming Specification. Version 2.3. NIST Interagency Report 7695. Gaithersburg: NIST,

2011. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/IR/nistir7695.pdf (дата звернення: 11.11.2025).

23. A Practical Guide to Common Platform Enumeration (CPE) // FOSSA Blog. 12.11.2025. URL: https://fossa.com/blog/practical-guide-common-platform-enumeration-cpe/ (дата звернення: 12.11.2025).

24. Python Software Foundation. smtplib – SMTP protocol client. Python 3 Standard Library Documentation. URL: https://docs.python.org/uk/3/library/smtplib.html (дата звернення: 19.11.2025).

25. Send email from a printer, scanner, or app // Google Workspace Admin Help. URL: https://support.google.com/a/answer/176600 (дата звернення: 20.11.2025).

26. smtplib: Tutorial with Code Snippets // Mailtrap Blog. 12.03.2025. URL: https://mailtrap.io/blog/smtplib/ (дата звернення: 29.11.2025).

27. Best practices for plain-text emails: a look at why they’re important // Litmus Blog. 31.08.2022. URL: https://www.litmus.com/blog/best-practices-for-plain-text-emails-a-look-at-why-theyre-important (дата звернення: 30.11.2025).

28. Python Software Foundation. venv – створення віртуальних середовищ. Документація Python 3. URL: https://docs.python.org/uk/3.9/library/venv.html (дата звернення: 04.12.2025).

29. Real Python. Python Virtual Environments: A Primer. URL: https://realpython.com/python-virtual-environments-a-primer/ (дата звернення: 04.12.2025).

30. Quickstart – Flask Documentation // Flask Documentation (3.1.x). URL: https://flask.palletsprojects.com/en/stable/quickstart/ (дата звернення: 06.12.2025).

31. Schedule a Python Script to Run Daily // GeeksforGeeks. 23.07.2025. URL: https://www.geeksforgeeks.org/python/schedule-a-python-script-to-run-daily/ (дата звернення: 08.12.2025).

32. International Software Testing Qualifications Board. Certified Tester Foundation Level Syllabus v4.0. Version 4.0.1. 15.09.2024. URL: https://istqb.org (дата звернення: 10.12.2025).

33. Graham D., Van Veenendaal E., Evans I., Black R. Foundations of Software Testing: ISTQB Certification. 4th ed. Cengage Learning EMEA, 2019. URL: https://www.cengage.com (дата звернення: 10.12.2025).

34. IEEE. IEEE Standard for Software and System Test Documentation (IEEE Std 829-2008). IEEE, 2008. URL: https://standards.ieee.org/standard/829-2008.html (дата звернення: 10.12.2025).

35. Закон України «Про охорону праці» від 14.10.1992 № 2694-XII. URL: https://zakon.rada.gov.ua/go/2694-12 (дата звернення: 15.12.2025).

36. НПАОП 0.00-7.15-18. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. URL: https://zakon.rada.gov.ua/go/z0508-18 (дата звернення: 15.12.2025)

37. ДСН 3.3.6.042-99. Державні санітарні норми мікроклімату виробничих приміщень : постанова Головного державного санітарного лікаря України від 01.12.1999 № 42. URL: https://zakon.rada.gov.ua/go/va042282-99 (дата звернення: 15.12.2025).

38. НАПБ А.01.001-2014. Правила пожежної безпеки в Україні. URL: https://zakon.rada.gov.ua/go/z0252-15 (дата звернення: 15.12.2025)

39. Кодекс цивільного захисту України : Закон України; Кодекс від 02.10.2012 № 5403-VI. URL: https://zakon.rada.gov.ua/go/5403-17 (дата звернення: 15.12.2025).

40. Стручок В. С. Безпека в надзвичайних ситуаціях. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної та заочної (дистанційної) форм навчання. Тернопіль : ФОП Паляниця В. А., 2022. 156 с. URL: http://elartu.tntu.edu.ua/handle/lib/39196 (дата звернення: 10.12.2025).

41. Стручок В. С. Техноекологія та цивільна безпека. Частина «Цивільна безпека». Навчальний посібник. Тернопіль : ТНТУ ім. І. Пулюя, 2022. 150 с. URL: http://elartu.tntu.edu.ua/handle/lib/39424 (дата звернення: 10.12.2025).

42. Михалик Д. М., Цуприк Г. Б., Бревус В. М., Мудрик І. Я., уклад. Методичні вказівки до виконання кваліфікаційної роботи магістра для здобувачів спеціальності 121 – Інженерія програмного забезпечення, всіх форм навчання

[Електронний ресурс]. Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2024. 44 с. URL: https://elartu.tntu.edu.ua/handle/lib/50316 (дата звернення: 02.12.2025).

43. Mudryk I., Petryk M. High-performance modeling, identification and analysis of heterogeneous abnormal neurological movement's parameters based on cognitive neuro feedback-influences [Електронний ресурс] // Innovative Solution in Modern Science. 2021. Vol. 1, Iss. 45. P. 235–249. URL: https://naukajournal.org/index.php/ISMSD/article/view/2386 (дата звернення: 03.12.2025).

44. Petryk M., Bachynskyi M., Brevus V., Mudryk I., Mykhalyk D. Analysis technology of neurological movements considering cognitive feedback influences of cerebral cortex signals [Електронний ресурс] // ITTAP. 2022. P. 45–54. URL: https://scholar.google.com/citations?view_op=view_citation&hl=uk&user=YIBK1fgAAAAJ&cstart=20&pagesize=80&sortby=pubdate&citation_for_view=YIBK1fgAAAAJ:QIV2ME_5wuYC (дата звернення: 03.12.2025).

ДОДАТКИ

ДОДАТОК А

Апробація результатів

УДК 004.056.5
В. Бурса, І. Мудрик
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ПЕРСПЕКТИВИ АВТОМАТИЗОВАНОГО ОПОВІЩЕННЯ ПРО ВРАЗЛИВОСТІ ПРОГРАМНИХ ПРОДУКТІВ НА ОСНОВІ ДАНИХ NVD ТА ІДЕНТИФІКАТОРІВ CPE

UDC 004.056.5
V. Bursa, I. Mudryk

PROSPECTS OF AUTOMATED VULNERABILITY ALERTING FOR SOFTWARE PRODUCTS BASED ON NVD DATA AND CPE IDENTIFIERS

Зважаючи на стрімке зростання кількості відомих вразливостей та постійне оновлення бази National Vulnerability Database, ручний моніторинг безпекових повідомлень стає малоефективним і створює ризик пропуску критичних загроз для програмних продуктів [1]. Навіть за наявності фільтрів у NVD спеціалісти змушені витрачати значний час на відбір релевантних записів.

Виникає питання: чи можна створити веб-сервіс, який автоматично відбиратиме потрібні записи з NVD на основі формального опису продуктів за допомогою Common Platform Enumeration та повідомлятиме фахівців про важливі вразливості електронною поштою [2]?

Метою роботи є проєктування веборієнтованої системи моніторингу, що періодично звертається до NVD, фільтрує вразливості за CPE-ідентифікаторами і ключовими словами та формує автоматичні e-mail-сповіщення для визначених отримувачів. Основна ідея полягає у налаштуванні користувачем «моніторингів», кожен з яких задає цільові продукти, ключові слова та пороговий рівень критичності; система регулярно виконує перевірку і надсилає стислий звіт із переліком актуальних CVE.

Подальший розвиток передбачає підключення додаткових джерел (вендорські бюлетені, списки активно експлуатованих вразливостей), розширення каналів сповіщення (месенджери, інтеграція з SIEM) та додавання аналітичних засобів для оцінки динаміки ризиків [3]. Це дозволить перетворити систему на гнучкий компонент комплексного процесу управління інформаційною безпекою.

Література

1. National Vulnerability Database (NVD). National Institute of Standards and Technology [Електронний ресурс]. – Режим доступу: https://nvd.nist.gov.
2. Cheikes B. A., Waltermire D., Scarfone K. Common Platform Enumeration: Naming Specification. Version 2.3 : NIST Interagency Report 7695 [Електронний ресурс]. – Gaithersburg, MD : NIST, 2011. – Режим доступу: https://csrc.nist.gov/pubs/ir/7695/final.
3. Кавун С. В., Носов В. В., Манжай О. В. Інформаційна безпека : навч. посіб. – Харків : Вид-во ХНЕУ, 2008. – 352 с.

ДОДАТОК Б

Структура проекту

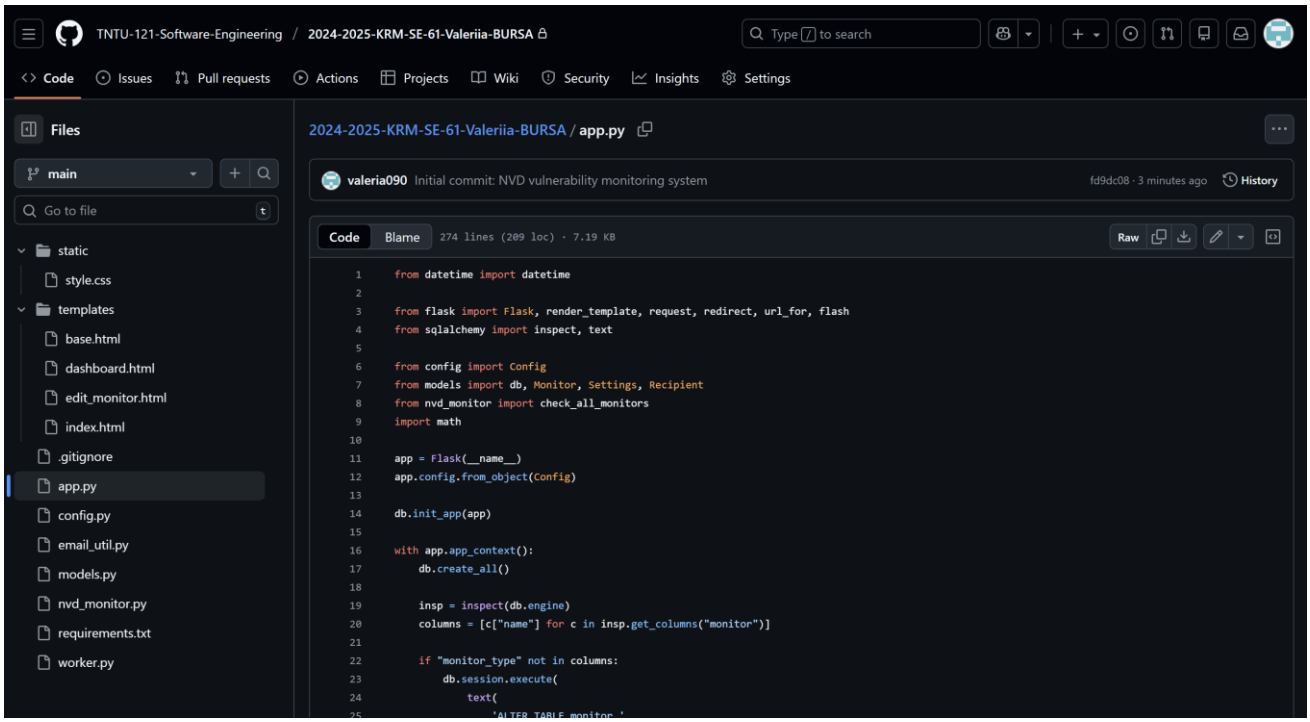

Рисунок Б.1 – Структура проекту на GitHub

Посилання на проект: https://github.com/TNTU-121-Software-Engineering/2024-2025-KRM-SE-61-Valeriia-BURSA