МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ І ПРИКЛАДНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЛАВРЕНТЮК НАЗАР ЮРІЙОВИЧ

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій
канд. техн. наук, доцент
_____ О. В. Зелінська
«_____» _____ 2024р.

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ ВИЯВЛЕННЯ ФІШИНГОВИХ АБО ШАХРАЙСЬКИХ ПОВІДОМЛЕНЬ У ЕЛЕКТРОННІЙ ПОШТІ КОРИСТУВАЧА

Спеціальність 122 Комп’ютерні науки

Кваліфікаційна (магістерська) робота

Науковий керівник:
Антонов Ю.С., доцент кафедри
інформаційних технологій,
канд. фіз.-мат. наук, доцент
_____
(Підпис)

Оцінка _____/_____/_____
(бали/за шкалою ЄКТС/за національною шкалою)

Голова ЕК:_____
(Підпис)

Вінниця 2024

Лаврентюк Н.Ю. Інформаційна система для виявлення фішингових або шахрайських повідомлень у електронній пошті користувача.

Дипломна робота викладена на 62 сторінці, містить 3 розділи, 9 ілюстрацій, 3 таблиці, 49 джерел.

Об’єктом розгляду є інформаційні системи для виявлення фішингових і шахрайських повідомлень у електронній пошті користувачів, зокрема, застосування телеграм-бота для цього процесу.

Предмет дослідження: методи виявлення фішингу та шахрайства в електронних листах за допомогою машинного навчання, аналізу тексту та чорних списків.

Мета роботи: розробити інформаційну систему для виявлення фішингових і шахрайських повідомлень в електронній пошті, що включає інтеграцію телеграм-бота як інтерфейсу для користувачів. Система має на меті застосовувати різні підходи для аналізу електронних листів і надавати користувачам інструменти для звітування про підозрілі повідомлення, а також для отримання рекомендацій щодо захисту від фішингу та шахрайства.

Перший розділ присвячено теоретичним основам виявлення фішингових і шахрайських повідомлень, а також принципам роботи з телеграм-ботами. У другому розділі розглянуто вимоги до інформаційної системи, а також принципи застосування машинного навчання та аналізу тексту для виявлення фішингу. Третій розділ присвячено безпосередній розробці системи, її архітектурі та інтерфейсу для користувачів.

У висновках обґрунтовано ефективність розробленої системи, а також запропоновано рекомендації щодо її впровадження та використання в реальних умовах.

Ключові слова: Інформаційна система, виявлення фішингу, шахрайство в електронній пошті, телеграм бот, машинне навчання, аналіз тексту, чорні списки, монолітна архітектура, мікросервісна архітектура, база даних, RESTful, API, Python, спам, фільтрація спаму, захист даних, кібербезпека, інтерфейс користувача, досвід користувача (UX).

Lavrentiuk N.Y. Information System for Detecting Phishing or Fraudulent Messages in User's E-Mail.

The thesis is presented on 62 pages, contains 3 chapters, 9 illustrations, 3 tables, and 49 sources.

The object of study is information systems for detecting phishing and fraudulent messages in users' emails, specifically the use of a Telegram bot for this process.

The subject of the research: methods for detecting phishing and fraud in emails using machine learning, text analysis, and blacklists.

The aim of the work: to develop an information system for detecting phishing and fraudulent messages in emails, which includes the integration of a Telegram bot as an interface for users. The system aims to apply various approaches to analyze emails and provide users with tools for reporting suspicious messages, as well as for receiving recommendations on how to protect themselves from phishing and fraud.

The first chapter is devoted to the theoretical foundations of detecting phishing and fraudulent messages, as well as the principles of working with Telegram bots. The second chapter discusses the requirements for the information system, as well as the principles of applying machine learning and text analysis to detect phishing. The third chapter focuses on the development of the system, its architecture, and the user interface.

The conclusions justify the effectiveness of the developed system, as well as provide recommendations for its implementation and use in real-life conditions.

Keywords: Information system, phishing detection, email fraud, Telegram bot, machine learning, text analysis, blacklists, monolithic architecture, microservice architecture, database, RESTful, API, Python, spam, spam filtering, data protection, cybersecurity, user interface, user experience (UX).

ЗМІСТ

ВСТУП

Актуальність роботи: З кожним днем все більше людей поринають у світ Інтернету. Це відкриває безліч нових можливостей, але й несе з собою певні ризики. На жаль, разом з зростанням кількості користувачів зростає й кількість шахраїв, які використовують довіру та незнання людей для власної вигоди. Саме тому системи, що зменшують ризики та підвищують користувацьку безпеку, стають все більш актуальними.

Мета дослідження: Розробити телеграм бота для виявлення фішингових або шахрайських повідомлень.

Завдання дослідження:

- аналіз предметної області;
- аналіз переваг та недоліків аналогів;
- розробити телеграм бота для виявлення фішингових або шахрайських повідомлень на основі отриманих результатів.

Об’єкт дослідження: виявлення фішингу, інтеграція систем.

Предмет дослідження: застосування Web-технологій для створення бота, який буде зменшувати кількість фішингу.

Практичне значення одержаних результатів: Це дозволить користувачам легко встановлювати та використовувати бота без залежності від інших платформ, а також бот може бути інтегрований з популярними поштовими клієнтами, такими як Gmail або Outlook, щоб автоматично перевіряти вхідні повідомлення на наявність фішингу та шахрайства.

Структура кваліфікаційної роботи: Бакалаврська робота складається із вступу, трьох розділів та висновків до них, списку використаних джерел та одного додатку.

У першому розділі бакалаврської роботи проведено огляд предметної області, та інструментів що дозволяють вирішити поставлену задачу.

У другому розділі наведено модель бота для виявлення фішингових або шахрайських повідомлень у електронній пошті користувача.

У третьому розділі наведено опис користувацького інтерфейсу, а також особливості налаштування технологій.

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМИ ВИЯВЛЕННЯ ФІШИНГОВИХ ТА ШАХРАЙСЬКИХ ПОВІДОМЛЕНЬ

1.1 Актуальність та масштаби проблеми фішингу та шахрайства в електронній пошті

1.1.1 Шкода від фішингу та шахрайства для користувачів та організацій [1]

Фішинг та шахрайство в електронній пошті є серйозною проблемою, яка може завдати значної шкоди як користувачам, так і організаціям. Потенційні наслідки цих злочинних дій включають [2]:

- **Фінансові втрати:** Фішингові атаки часто спрямовані на крадіжку особистої інформації, такої як номери банківських рахунків, паролі та дані кредитних карток, яку потім можна використовувати для здійснення несанкціонованих транзакцій або крадіжки коштів.
- **Крадіжка особистих даних[3]:** Шахрайські листи можуть використовуватися для збору особистої інформації, такої як адреси, номери телефонів, дати народження, яку потім можна використовувати для ідентифікаційної крадіжки або інших шахрайських цілей.
- **Репутаційні ризики:** Фішингові атаки можуть бути спрямовані на дискредитацію репутації людей або організацій, поширюючи дезінформацію або шкідливий контент.
- **Втрата продуктивності:** Фішингові та шахрайські листи можуть завантажувати комп'ютери користувачів шкідливим програмним забезпеченням, яке може призвести до втрати даних, порушення роботи систем та зниження продуктивності.
- **Психологічний вплив[4]:** Жертви фішингу та шахрайства можуть відчувати стрес, тривогу та гнів, а також втратити довіру до онлайн-комунікацій.

1.1.2 Зростання кількості та складності фішингових атак

За даними досліджень, кількість та складність фішингових атак постійно зростають. Зловмисники використовують все більш витончені методи для обману користувачів, роблячи свої листи та веб-сайти більш реалістичними та переконливими.

Ось деякі з факторів, які сприяють зростанню проблеми [5]:

– Широке використання електронної пошти: Електронна пошта є одним з найпоширеніших каналів онлайн-спілкування, що робить її ідеальною мішенню для фішингових атак.

– Зростання онлайн-платежів: Зростаюча популярність онлайн-платежів робить користувачів більш схильними до фішингових атак, які націлені на крадіжку їхніх платіжних даних.

– Розвиток технологій: Зловмисники використовують нові технології, такі як штучний інтелект та машинне навчання, для створення більш персоналізованих та цільових фішингових атак.

– Недостатня обізнаність користувачів: Багато користувачів не знають про ризики фішингу та шахрайства в електронній пошті, що робить їх більш вразливими до цих атак.

Враховуючи значний ризик та зростаючу складність фішингових атак, розробка ефективних методів виявлення та запобігання цим злочинним діям є вкрай актуальною.

1.2 Існуючі методи виявлення фішингових та шахрайських повідомлень

Існує два основних типи методів виявлення фішингових та шахрайських повідомлень в електронній пошті, а саме методи засновані на правилах, та методи, які ґрунтуються на машинному навчанні. Кожен з них має свої переваги та недоліки, тому розберемо їх детальніше.

1.2.1 Традиційні методи, засновані на правилах

Ці методи ґрунтуються на наборі визначених правил, які описують

характеристики фішингових та шахрайських повідомлень. Ці правила можуть включати[6]:

– Наявність підозрілих URL-адрес: Фішингові листи часто містять URL-адреси, які ведуть на фейкові веб-сайти, що імітують легітимні веб-сайти.

– Використання підозрілих слів або фраз: Фішингові та шахрайські листи часто містять підозрілі слова або фрази, які намагаються спонукати користувачів розкрити особисту інформацію або натиснути на шкідливі посилання.

– Відсутність контактної інформації: Легітимні компанії зазвичай надають контактну інформацію, таку як адреса електронної пошти, номер телефону та фізична адреса, у своїх повідомленнях. Фішингові листи часто не містять такої інформації.

– Непрофесійне оформлення: Фішингові листи часто мають непрофесійне оформлення, з помилками в граматиці, орфографії та форматуванні.

Традиційні методи, засновані на правилах[7], є простими та ефективними для виявлення деяких очевидних фішингових та шахрайських повідомлень. Однак ці методи мають певні обмеження:

– Нездатність виявляти нові та складні фішингові атаки: Зловмисники постійно вдосконалюють свої методи, тому традиційні правила можуть бути не в змозі виявити нові та складні фішингові атаки.
– Високий рівень помилок: Традиційні правила можуть помилково позначати легітимні повідомлення як фішингові, що призводить до незручностей для користувачів.

1.2.2 Методи, що ґрунтуються на машинному навчанні

Методи, що ґрунтуються на машинному навчанні, використовують статистичні моделі для аналізу великих наборів даних електронних листів, щоб виявити закономірності, які відрізняють фішингові та шахрайські повідомлення від легітимних. Ці методи можуть використовувати різні алгоритми машинного навчання, такі як [8]:

- Наївний алгоритм Байєса: цей алгоритм класифікує повідомлення на основі ймовірності того, що вони містять певні характеристики, пов'язані з фішингом або шахрайством.
- Підтримка векторів (SVM): цей алгоритм створює гіперплощину, яка відокремлює фішингові та шахрайські повідомлення від легітимних.
- Дерева рішень: ці алгоритми приймають серію рішень на основі характеристик повідомлення, щоб визначити, чи є воно фішинговим або шахрайським.

Методи, що ґрунтуються на машинному навчанні, мають ряд переваг перед традиційними методами:

- Здатність виявляти нові та складні фішингові атаки: машинне навчання може адаптуватися до нових даних, що робить його більш ефективним для виявлення нових та складних фішингових атак.
- Низький рівень помилок: методи машинного навчання можуть бути навчені на великих наборах даних, щоб мінімізувати ймовірність помилкової класифікації легітимних повідомлень як фішингових.

Однак методи, що ґрунтуються на машинному навчанні, також мають певні недоліки[9]:

- Складність реалізації: розробка та навчання моделей машинного навчання може бути складним та трудомістким процесом. Це потребує знань в області машинного навчання, доступу до потужних обчислювальних ресурсів та навичок роботи з великими наборами даних.
- Вимоги до даних: для навчання моделей машинного навчання потрібні великі набори даних з чітко позначеними фішинговими та легітимними повідомленнями. Збирання та маркування таких даних може бути дорогим та затратним по часу.
- Прозорість: моделі машинного навчання можуть бути складними для розуміння, що ускладнює інтерпретацію їхніх результатів та пояснення причин, чому певні повідомлення були класифіковані як фішингові або

легітимні. Це може призвести до проблем з довірою та прийняттям рішень.

- Уразливість до атак: моделі машинного навчання можуть бути вразливими до атак, таких як атаки зворотного навчання або атаки отруєння даних. Зловмисники можуть використовувати ці атаки, щоб маніпулювати моделями та змусити їх помилково класифікувати повідомлення.

1.2.3 Порівняльний аналіз існуючих методів

Вибір методу виявлення фішингу та шахрайства залежить від конкретних потреб та ресурсів.

Традиційні методи можуть бути достатніми для організацій з невеликим обсягом електронної пошти та обмеженими ресурсами.

Однак для організацій з великим обсягом електронної пошти та високими вимогами до точності та надійності рекомендується використовувати методи, що ґрунтуються на машинному навчанні.

Важливо зазначити, що жоден метод не є ідеальним, і завжди існує ризик того, що деякі фішингові або шахрайські повідомлення можуть залишитися непоміченими.

Тому важливо використовувати комбінацію методів та постійно оновлювати та вдосконалювати системи виявлення для протидії новим та складним загрозам.

1.3 Сервіси-аналоги

При пошуку сервісів-аналогів неможливо не згадати антивірусні застосунки, які на сьогоднішній день доступні у багатьох видах, таких як застосунки, веб-сайти та розширення для браузерів.

Можна розглянути потенціал деяких існуючих програм і сервісів, які частково або повністю вирішують подібні проблеми:

Kaspersky Internet Security [10]: Даний застосунок дає певний захист від фішингових атак, а саме:

- **Антифішинговий захист**: блокує доступ до фішингових веб-сайтів.
- **Захист від онлайн-шахрайства**: Програма попереджає користувачів про

підозрілі веб-сайти та електронні листи, які можуть використовуватися для шахрайських цілей.

- **Захист від шкідливого програмного забезпечення**: Kaspersky Internet Security також захищає комп'ютер від вірусів, троянів та іншого шкідливого програмного забезпечення.

Norton 360 [11]: Чи не найкращий сервіс для прямої перевірки на наявність фішингу, надає такі інструменти як:

- Аналіз URL-адрес: Norton 360 перевіряє посилання в електронних листах на шкідливі веб-сайти.
- Аналіз тексту: Програма сканує текст повідомлень, щоб знайти слова та фрази, які можуть бути використані в фішингових атаках.
- Аналіз зображень: Norton 360 може виявити шкідливі фотографії, які можуть містити шкідливий код або посилання.

Недоліком цих застосунків є доступність, адже щоб використовувати повний функціонал потрібно проплачувати підписку, а також попередньо завантажувати застосунок.

PhishingSecure [12]: Це комплексне рішення для надійного захисту від фішингу включає:

- Фішинговий фільтр: PhishingSecure перевіряє електронні листи на фішингові посилання та шкідливий контент.
- Управління користувачами: Адміністратори мають можливість змінювати політики та контролювати, як користувачі можуть отримати доступ до сервісу.
- Навчання користувачів: PhishingSecure надає користувачам навчальні матеріали, щоб допомогти їм визначити та уникати фішингових атак.

Proofpoint Email Protection [13]: Захист електронної пошти Proofpoint використовує штучний інтелект і машинне навчання для виявлення фішингових атак. Гарантія електронної пошти від Proofpoint включає:

- Багаторівневий аналіз: сервіс розпізнає фішингові повідомлення за

допомогою аналізу URL-адрес, текстів, зображень та інших елементів електронних листів.

- Захист від нових небезпек: захист електронної пошти Proofpoint постійно оновлюється, щоб протистояти новим і складним фішинговим атакам.
- Звіти та аналітика: Сервіс надає детальні звіти про фішингові атаки, що дозволяє адміністраторам оцінювати ризики та вдосконалювати методи захисту.

Також останнім часом зростає популярність розширень для браузера, адже частіше всього фішингові посилання знаходяться сам у скріптах сайта чи повідомленнях у поштах, тому такі інструменти дуже зручні і часто підвищують вашу безпеку. Ось декілька прикладів таких застосунків:

Anti-Phishing Tool: це безкоштовне розширення для Chrome, яке блокує доступ до веб-сайтів, спрямованих на фішинг.

NoScript [14]: розширення для браузерів Chrome та Firefox, яке блокує JavaScript на веб-сайтах автоматично. Враховуючи те, що JavaScript часто використовується в фішингових атаках, NoScript може допомогти користувачам залишатися захищеними.

Також варті уваги саме телеграм боти, які допомагають захиститись як у телеграмі, адже часто люди надсилають фішингові посилання, так і у пошті через gmail API [15]:

– **@phish_bot:** Цей бот від Anti-Phishing Working Group (APWG) дозволяє користувачам повідомляти про підозрілі веб-сайти та посилання. Бот потім аналізує повідомлення та додає їх до чорного списку, якщо вони визнані шкідливими.

– **@stop_phishing_bot:** Цей бот від Stop Phishing дозволяє користувачам перевіряти URL-адреси на наявність фішингових веб-сайтів. Бот використовує базу даних шкідливих веб-сайтів для визначення, чи є URL-адреса безпечною для відвідування.

– **@antiphishing_bot:** Цей бот від Anti-Phishing Alliance дозволяє

користувачам повідомляти про підозрілі веб-сайти та посилання. Бот потім аналізує повідомлення та додає їх до чорного списку, якщо вони визнані шкідливими .

– **@phishing_awareness_bot:** Цей бот від Phishing Awareness Training Platform навчає користувачів про фішинг та те, як його розпізнати. Бот пропонує вікторини, поради та інші навчальні матеріали.

– **@PhishingReportBot:** Цей бот від Phishing Report дозволяє користувачам повідомляти про підозрілі веб-сайти та посилання. Бот аналізує повідомлення та додає їх до чорного списку, якщо вони визнані шкідливими.

Звісно, важливо зазначити, що усі дані застосунки, боти тощо не є стовідсотковою гарантією безпеки. Завжди потрібно додатково перевіряти і відноситись з обережністю до невідомих посилань та повідомлень, навіть якщо застосунки говорять, що усе безпечно.

1.4 Цілі інформаційної системи для виявлення фішингових та шахрайських повідомлень.

1.4.1 Функціональні цілі

Основною метою системи є надання автоматизованого механізму для перевірки електронних листів на предмет фішингових і шахрайських повідомлень. Система повинна забезпечити детальний аналіз текстового вмісту, URL-адрес, графічних елементів та інших складових електронних повідомлень.

Однією з ключових завдань є ідентифікація спроб імітації легітимних веб-сайтів чи організацій, виявлення повідомлень зі шкідливими посиланнями або вкладеннями, а також розпізнавання спроб маніпуляції користувачами з метою отримання їх особистих даних.

Завданням системи[16] є класифікація електронних листів за категоріями: фішингові, шахрайські або легітимні. Для досягнення цієї мети система має використовувати такі методи:

- Розробка алгоритмів машинного навчання.
- Створення правил на основі ключових слів.

- Використання чорних та білих списків URL-адрес.

Іншою важливою ціллю є інформування користувачів про підозрілі електронні листи, що може включати:

- Автоматичне переміщення підозрілих листів у спеціальну папку.
- Маркування підозрілих листів спеціальними позначками.
- Відправлення користувачам повідомлень про потенційну небезпеку фішингу чи шахрайства.

Система також повинна забезпечувати управління користувачами та їхніми налаштуваннями, включаючи:

- Управління доступом користувачів.
- Налаштування параметрів чутливості детектування фішингу.
- Можливість для користувачів самостійно маркувати листи.

Необхідною є також функціональність звітування та логування для оцінки ефективності системи.

1.4.2. Нефункціональні цілі

Для забезпечення надійної роботи системи, основним завданням є підтримання високого рівня продуктивності та безпеки. Потрібно звернути увагу на кожен з базових аспектів.

Продуктивність:

- Забезпечення здатності системи обробляти високі обсяги електронної пошти без зниження продуктивності.
- Гарантування, що користувачі не зіткнуться з затримками при обробці їхньої електронної пошти.

Масштабованість:

Система повинна бути масштабована, щоб підтримувати більшу кількість користувачів і електронних листів.

Це надзвичайно важливо, оскільки система повинна бути в змозі підтримувати зростання кількості користувачів, а також кількості електронних листів, які надсилаються.

Безпека[17]:

Щоб захистити персональні дані користувачів, система має виконувати базові правила безпеки:

- Запровадження шифрування даних користувачів для захисту їхньої конфіденційності.
- Захист системи від несанкціонованого доступу та інших загроз.
- Постійне оновлення системи для виправлення вразливостей.

Надійність:

Система повинна бути надійною, щоб вона працювала безперебійно.

Це важливо, тому що користувачі повинні мати можливість покладатися на систему для захисту їх від фішингових та шахрайських повідомлень.

Простота використання:

Система повинна бути простою у використанні для користувачів.

Користувачі не повинні потребувати спеціальних знань або навичок для використання системи.

1.5 Висновки до розділу 1

Проведений аналіз сучасного стану проблеми фішингу та шахрайства в електронній пошті дозволив виявити актуальність дослідження та обґрунтувати необхідність розробки ефективних методів захисту від таких загроз. Було проаналізовано існуючі методи виявлення фішингових повідомлень, їхні переваги та недоліки. Особливу увагу було приділено різноманітності сучасних фішингових атак, які постійно ускладнюються та адаптуються до нових умов. На основі проведеного аналізу визначені ключові напрямки для подальшого дослідження та розробки системи захисту від фішингу на основі Telegram-бота, яка зможе ефективно протистояти як традиційним, так і новим видам атак.

РОЗДІЛ 2
ПОБУДОВА МОДЕЛІ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ВИЯВЛЕННЯ ФІШИНГОВИХ ТА ШАХРАЙСЬКИХ ПОВІДОМЛЕНЬ В ЕЛЕКТРОННІЙ ПОШТІ

2.1 Загальна архітектура

Загальна архітектура інформаційної системи для виявлення фішингових або шахрайських повідомлень у електронній пошті користувача складається з кількох основних компонентів, які взаємодіють між собою для забезпечення ефективного аналізу та обробки повідомлень. Система включає базу даних, різноманітні API для інтеграції, логіку обробки та аналізу повідомлень, а також інтерфейс взаємодії з користувачем через Telegram бота.

Опис компонентів системи

1. База даних:
   - Зберігає інформацію про користувачів, отримані повідомлення, результати аналізу та інші необхідні дані.
   - Забезпечує швидкий доступ до даних для обробки та аналізу.
2. API інтеграції
   - Gmail API[18]: Використовується для доступу до електронної пошти користувача, отримання повідомлень для аналізу.
   - SafeBrowsing API[19]: Використовується для перевірки URL-адрес у повідомленнях на наявність фішингових або шахрайських сайтів.
   - Telegram Bot API[20]: Використовується для створення бота, який взаємодіє з користувачами, надає результати аналізу та іншу інформацію.
3. Логіка обробки та аналізу повідомлень[21]
   - Включає алгоритми для аналізу вмісту повідомлень, виявлення підозрілих елементів, таких як фішингові URL-адреси, шкідливі вкладення тощо.

- Використовує результати перевірки SafeBrowsing API для визначення небезпечних посилань.
- Обробляє дані з бази даних для створення звітів та сповіщень.

4. Інтерфейс взаємодії з користувачем
   - Реалізований через Telegram бота, який надає користувачам можливість отримувати результати аналізу повідомлень, запитувати додаткову інформацію та взаємодіяти з системою.

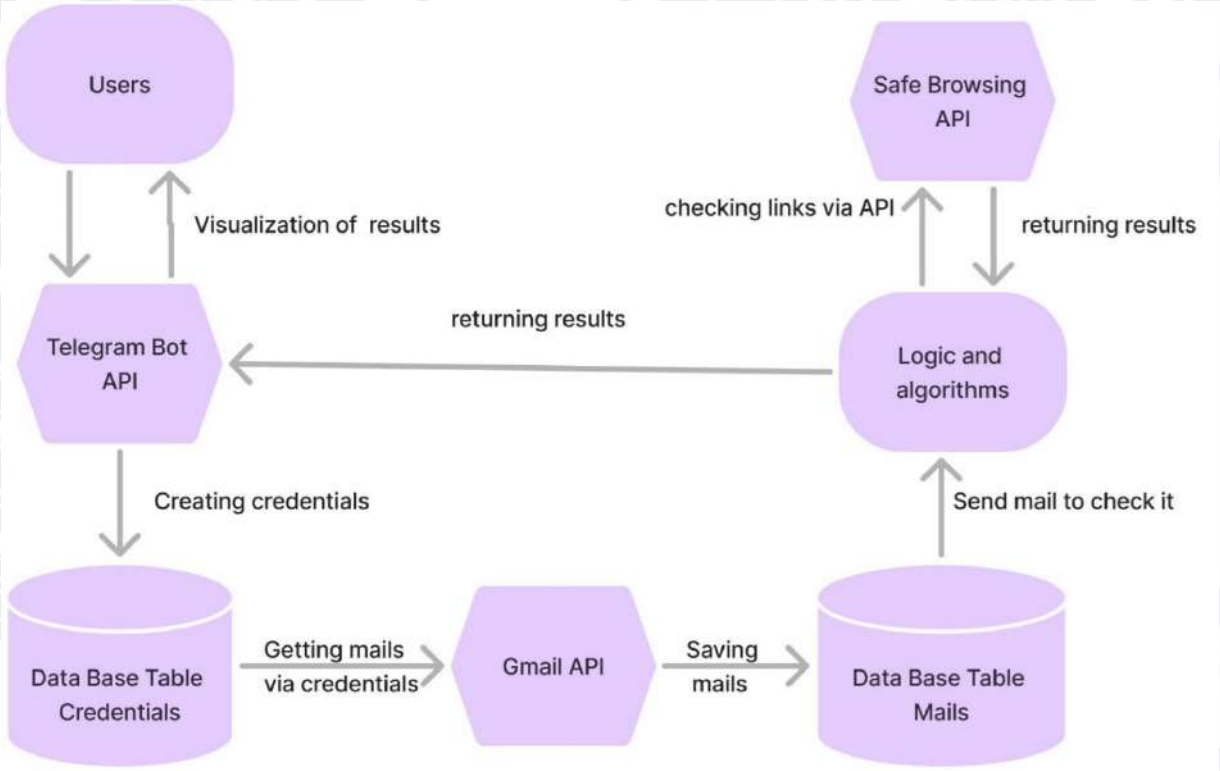


Рис 2.1. Схема Архітектури

2.2 Аналіз вимог до бота в Telegram

Перед тим як приступити до створення бота в Telegram, який відповідає за детекцію фішингових та шахрайських активностей у електронних листах користувачів [22], необхідно чітко визначити набір правил і стандартів. Ці критерії слугуватимуть основою для гладкого та ефективного процесу розробки, забезпечуючи, щоб усі аспекти проєкту були враховані з самого початку. Тому встановимо вимоги для бота для кращого розуміння конкретності розробки.

Функціональні вимоги до бота[23]:

- Інтеграція з електронною поштою: Бот повинен мати постійну можливість підключатися до поштової скриньки користувача та перехоплювати вхідні повідомлення для подальшого аналізу.
- Виявлення фішингових повідомлень: Бот має використовувати сучасні алгоритми для аналізу та ідентифікації потенційно фішингових або шахрайських елементів у повідомленнях.
- Класифікація повідомлень: Для зручності бот повинен мати змогу ділити повідомлення на легітимні, підозрілі або фішингові, з відповідними позначками для користувачів.
- Взаємодія з користувачем: Команди бота для управління налаштуваннями та перегляду звітів про виявлені фішингові повідомлення повинні бути інтуїтивно зрозумілі.
- Звітування: Бот матиме змогу надсилати звіти або сповіщення про виконану роботу.

Нефункціональні вимоги та обмеження:

- Безпека[24]: Забезпечення конфіденційності даних користувачів та захист від несанкціонованого доступу.
- Продуктивність: Бот повинен обробляти повідомлення швидко та ефективно і мінімізувати затримки у взаємодії з користувачами.
- Масштабованість[25]: Система повинна мати можливість масштабуватися зі збільшенням кількості користувачів.
- Надійність: Бот має працювати надійно, з мінімальною кількістю помилок і збоїв.
- Відповідність нормам і правилам: Бот повинен відповідати всім відповідним законодавчим нормам, включаючи GDPR.

Очікувані сценарії використання:

- Реєстрація та налаштування: Користувачі повинні мати можливість легко реєструвати та налаштовувати бота через Telegram.

– Повсякденне використання: Бот повинен автоматично сканувати нові повідомлення та повідомляти користувачів про будь-які підозрілі знахідки.
– Взаємодія з користувачем при виявленні фішингу: Якщо виявлено фішингове повідомлення, користувач повинен отримати сповіщення або звіт з рекомендаціями щодо подальших дій.
– Звітність та аналіз[26]: Користувачі у будь який момент часу повинні мати змогу переглядати звіти та статистику щодо діяльності бота.

Цей аналіз вимог є важливою частиною у процесі розробки, оскільки він формує основу для проєктування архітектури системи та подальшого вибору технологій.

2.3 Проєктування архітектури бота

Проєктування архітектури бота - важливий етап, що визначає, як буде побудована система загалом і як взаємодіятимуть її компоненти. Щоб забезпечити ефективність і масштабованість бота Telegram, найкраще буде використовувати модульний підхід, який дає змогу легко розширювати й оновлювати систему.

Основні компоненти архітектури:

– Інтерфейс користувача (UI) [27]: Містить Telegram-бота і надає користувачам інтерфейс для взаємодії з системою.

– Модуль обробки повідомлень: Відповідає за приймання та аналіз повідомлень від користувачів і надсилання реакцій та відповідей.

– Модуль аналізу даних: Використовує алгоритми машинного навчання та евристику для виявлення фішингових і шахрайських повідомлень.

– База даних[28]: Зберігає інформацію про користувачів, їх налаштування, історію повідомлень і результати аналізу.

– Модуль інтеграції з електронною поштою[29]: Забезпечує взаємодію з поштовою скринькою користувача: приймає та аналізує електронні листи.

– Система управління та моніторингу: Відстежує стан системи, фіксує події та надає інструменти для адміністрування.

Розробляючи архітектуру бота для виявлення фішингових і шахрайських повідомлень у Telegram, хотілось створити систему, яка була б не тільки ефективною та надійною, а й гнучкою та масштабованою. В основі архітектури лежить модульний підхід, що дозволяє кожній частині системи працювати індивідуально, але синхронно з іншими компонентами.

Центральне місце в архітектурі посідає користувацький інтерфейс, представлений ботом Telegram, який забезпечує зручну взаємодію між користувачем і системою. Цей інтерфейс пов'язаний з модулем обробки повідомлень, який аналізує і реагує на вхідні дані від користувача. Для виявлення потенційно небезпечного контенту в повідомленнях модуль аналізу даних використовує передові алгоритми машинного навчання та евристики.

Зберігання всієї інформації, включно з даними користувача, його уподобаннями, історією повідомлень і результатами аналізу, покладено на захищену базу даних. Ця база даних є основоположним компонентом, що забезпечує центральне сховище всієї важливої інформації.

Важливим елементом архітектури є також модуль інтеграції з електронною поштою, який дає змогу ботам взаємодіяти з поштовими скриньками користувачів, отримувати й аналізувати листи на предмет шахрайства. Це значно розширює можливості бота і робить його корисним інструментом у боротьбі з фішингом.

Для забезпечення стабільного функціонування та високої доступності системи використовується система управління та моніторингу. Ця система відстежує стан усіх компонентів системи, фіксує події та надає інструменти для управління.

Під час проєктування архітектури було дотримано принципів модульності для спрощення розроблення та тестування системи, масштабованості для витримування зростаючих навантажень, високої доступності для забезпечення безперервної роботи та безпеки для захисту користувацьких даних.

Використання шаблонів проєктування, таких як мікросервіси, дозволяє

нам створити гнучку та ефективну систему, яка може легко адаптуватися до змінних потреб користувачів та технологічних умов. Ці шаблони також сприяють асинхронній обробці даних, зменшуючи навантаження на систему та покращуючи загальну продуктивність.

Таким чином, ретельно продумані архітектури ботів є основою для створення надійних і безпечних інструментів, які допомагають користувачам уникнути фішингу та шахрайства під час спілкування електронною поштою.

2.4 Розробка алгоритмів для виявлення фішингу

Хоча виявлення фішингових повідомлень - складне завдання, що вимагає аналізу великих обсягів даних і розроблення складних алгоритмів, які дають змогу виявляти потенційні загрози, проаналізувавши весь процес фішингових атак, можна зробити висновок, що найлегше помітити фішинг можна саме у розсилці (рис. 2.2).

Рис. 2.2. Схема послідовності фішингової атаки

Процес розробки таких алгоритмів включає декілька ключових етапів[30]:

1) Збір даних: Цей крок передбачає створення бази даних, що містить як фішингові, так і легітимні повідомлення. Фішингові повідомлення можна збирати з різних джерел, зокрема з приватних баз даних, антифішингових сервісів та публічних джерел. Легітимні повідомлення можуть бути зібрані з відкритих джерел або надані користувачами за їхньою згодою.
2) Попередня обробка даних: Повідомлення повинні бути очищені та

стандартизовані, перш ніж вони будуть використані для навчання алгоритмів. Це включає видалення непотрібних символів, перетворення тексту в нижній регістр, видалення стоп-слів, та використання стемінгу.

3) Вибір ознак: Ознаки, які використовуються для класифікації повідомлень, мають велике значення. Вони можуть включати текстові характеристики, метадані, структуру листа, наявність певних URL, тощо.
4) Розробка моделі машинного навчання: На цьому етапі розробники використовують різні алгоритми машинного навчання, такі як логістична регресія[31], нейронні мережі, або ансамблеві методи[32], для створення моделі, здатної розпізнавати фішинг.
5) Тренування та оптимізація моделі: Модель тренується на заздалегідь визначеному наборі даних, після чого проводиться її оптимізація для підвищення точності та зниження помилок.
6) Валідація та тестування: Перевірка моделі на незалежному наборі даних дозволяє оцінити її ефективність. Це включає в себе вимірювання точності, повноти та інших метрик якості.
7) Імплементація евристичних методів: Для підвищення точності виявлення можуть бути використані евристичні правила, засновані на специфічних шаблонах поведінки фішингових атак.
8) Інтеграція алгоритму: Готовий алгоритм інтегрується у відповідні системи та додатки для виявлення фішингових повідомлень в режимі реального часу.
9) Моніторинг та оновлення: Після впровадження алгоритму його необхідно регулярно оновлювати, щоб він міг адаптуватися до нових видів фішингових атак.

Після ретельного проходження через кожен з ключових етапів, від початку дослідження до розробки та перевірки моделі, наступним кроком є інтеграція алгоритму в реальні системи. Цей процес слід здійснювати з додатковою обережністю, адже від точності та надійності алгоритму залежить безпека

користувачів. На цьому етапі важливим є забезпечення того, щоб алгоритм не лише виявляв фішингові повідомлення з високою точністю, а й щоб він був здатний швидко адаптуватися до нових видів атак, які постійно з'являються в результаті постійної еволюції методів шахрайства.

Окрім того, алгоритм повинен бути вбудований таким чином, щоб його можна було легко оновлювати та масштабувати відповідно до зростаючих потреб та обсягів даних. Це вимагає глибокого розуміння не тільки самої технології, а й специфіки домену, в якому вона застосовується. Така інтеграція вимагає тісної співпраці між розробниками, аналітиками даних та кінцевими користувачами, щоб забезпечити максимальну ефективність та користь від впровадження алгоритму.

Кінцевою метою цього процесу є створення надійного та ефективного інструменту, який буде слугувати надійним щитом проти фішингових атак, мінімізуючи ризики для користувачів та організацій. Такий інструмент зможе не тільки зменшити кількість успішних фішингових атак, але й підвищити обізнаність користувачів щодо цієї проблеми, сприяючи більш безпечному цифровому середовищу.

2.5 Інтеграція бота з електронною поштою

Інтеграція бота для виявлення фішингових повідомлень у систему електронної пошти є ключовим кроком, який забезпечує реальне використання розробленого рішення. Цей процес включає кілька важливих аспектів, таких як забезпечення сумісності з існуючими електронними поштовими клієнтами, забезпечення безпеки даних користувачів, а також забезпечення мінімального впливу на продуктивність системи пошти.

Під час інтеграції бот повинен бути налаштований таким чином, щоб він міг аналізувати вхідні повідомлення на предмет фішингу, використовуючи розроблені алгоритми, і надавати користувачам відповідні попередження (рис. 2.2). Це вимагає взаємодії з поштовим сервером для перехоплення повідомлень, їх аналізу та відповідного реагування у випадку виявлення підозрілої активності.

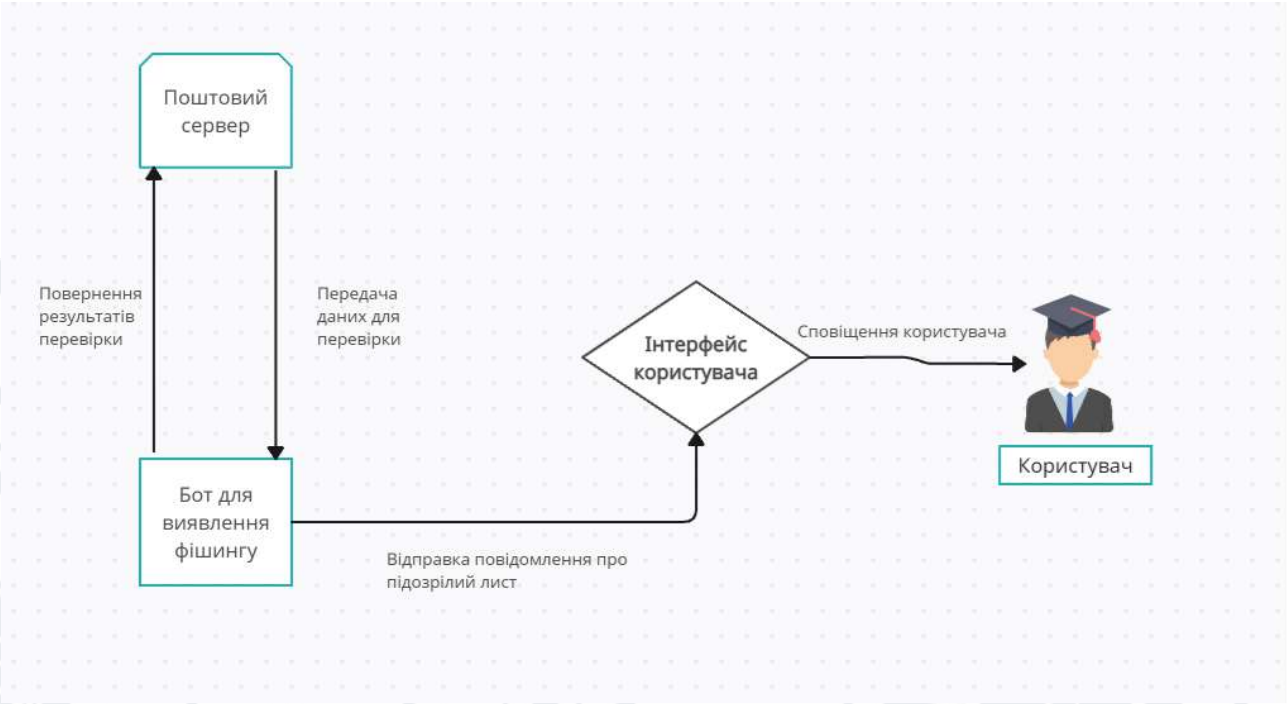


Рис. 2.3 Схема взаємодії користувача та бота

Перш за все, необхідно розробити архітектуру інтеграції, яка враховує всі аспекти поштової служби, включаючи протоколи зв'язку, безпеку даних та збереження конфіденційності. Бот повинен бути розроблений так, щоб його можна було легко встановити на сервері, не вимагаючи значних змін в існуючій інфраструктурі.

Процес інтеграції складається з кількох кроків:

– Аутентифікація та авторизація[33]: Бот має бути належним чином авторизований для доступу до поштових повідомлень без порушення приватності користувачів.

– Перехоплення повідомлень: Після авторизації бот повинен мати змогу перехоплювати вхідні та вихідні повідомлення для аналізу без затримок в доставці.

– Аналіз даних: Використовуючи попередньо навчені моделі машинного навчання або евристичні правила, бот аналізує зміст повідомлень на наявність фішингових ознак.

– Відповідь системи[34]: Якщо фішинг виявлено, бот вживає попередньо визначених заходів, наприклад, переміщає повідомлення до папки "Спам", позначає його як підозріле або навіть блокує відправника.

– Сповіщення користувачів: Користувачі повинні бути повідомлені про виявлені фішингові атаки відповідним і зрозумілим способом, щоб знизити ризик їх взаємодії з шкідливими повідомленнями.

– Логування та аудит: Всі дії бота повинні бути зареєстровані для подальшого аналізу та вдосконалення системи.

Після впровадження бота важливо продовжувати моніторинг його роботи та реагувати на нові загрози шляхом оновлення алгоритмів та баз даних. Це забезпечить, що система залишатиметься актуальною та ефективною у боротьбі з фішингом.

Завдяки цим крокам інтеграції, бот для виявлення фішингу може стати невід'ємною частиною інфраструктури електронної пошти, забезпечуючи високий рівень захисту для користувачів без значного втручання в їх повсякденне використання електронної пошти.

2.6 Взаємодія користувача з ботом

Взаємодія користувача з ботом для виявлення фішингу є ключовим аспектом, який впливає на успішність захисту від кібератак[35]. Для ефективної роботи системи важливо, щоб користувачі могли легко і зручно розуміти та управляти повідомленнями, позначеними як підозрілі. Нижче наведено основні принципи взаємодії користувачів з ботом:

– **Сповіщення:** Коли бот виявляє фішинг, він повинен надсилати користувачу зрозумілі та невідкладні сповіщення. Ці сповіщення можуть бути у вигляді візуальних сигналів у користувацькому інтерфейсі поштової скриньки або через інші канали комунікації, залежно від налаштувань користувача.

– **Інтерфейс користувача:** Інтерфейс, через який користувачі взаємодіють з ботом, має бути інтуїтивно зрозумілим і лаконічним,

дозволяючи користувачам швидко зрозуміти причини, за якими повідомлення було позначено як фішинг, та вибрати відповідні дії.

– **Навчання користувачів:** Бот може включати елементи, які навчають користувачів розпізнавати фішинг, надаючи інформацію та поради щодо кібербезпеки, тим самим підвищуючи рівень обізнаності користувачів.

– **Взаємодія з ботом:** Користувачі повинні мати можливість легко надати зворотний зв'язок боту, повідомляючи про помилкові спрацьовування або пропущені фішингові спроби, що допоможе поліпшити точність виявлення.

– **Зворотний зв'язок та адаптація:** Бот має бути здатним до самонавчання, використовуючи зворотний зв'язок від користувачів для налаштування своїх алгоритмів та покращення роботи.

– **Прозорість та контроль[36]:** Користувачам слід надати чітке розуміння того, як бот обробляє їхні дані, а також контроль над налаштуваннями захисту, включаючи можливість відключення певних функцій бота.

Забезпечення зручної та ефективної взаємодії між користувачем та ботом знижує ризики, пов'язані з фішингом, та допомагає підтримувати безпечне цифрове середовище для користувачів.

2.6 Тестування бота

Тестування бота для виявлення фішингу є важливим етапом у процесі розробки, що дозволяє забезпечити його надійність та ефективність перед впровадженням у реальні умови експлуатації. Нижче наведено ключові компоненти стратегії тестування[37]:

– **Стратегія тестування:**

Стратегія тестування повинна включати як автоматизовані, так і ручні методи перевірки, а також різні типи тестування, такі як модульне тестування, інтеграційне тестування, системне тестування та приймальне тестування. Тестування повинно охоплювати різні варіанти використання, включаючи реальні середовища, і бути спрямоване на

виявлення потенційних помилок, проблем з дотриманням вимог та інших вразливостей.

- **Автоматизоване та ручне тестування:**

  Автоматизоване тестування може використовувати заздалегідь визначені тестові сценарії для швидкого сканування великих обсягів електронної пошти на предмет фішингу. Це дозволяє ефективно виявляти систематичні помилки та недоліки в алгоритмах. З іншого боку, ручне тестування дозволяє тестувальникам аналізувати поведінку ботів у конкретних ситуаціях, де потрібне людське судження, наприклад, для оцінки адекватності реакції бота на складну фішингову атаку.

- **Валідація точності виявлення фішингових повідомлень:**

  Точність виявлення фішингових повідомлень є критично важливим аспектом, який потребує ретельної валідації. Вона включає аналіз таких показників, як частота помилково позитивних (false positives) та помилково негативних (false negatives) результатів. Використання різноманітного набору даних з реальними та симульованими фішинговими атаками дозволяє всебічно оцінити ефективність бота. Також важливо тестувати бота на здатність адаптуватися до нових типів фішингових схем, що постійно еволюціонують.

2.7 Розгортання та моніторинг бота

Розгортання бота для виявлення фішингу[38] починається з вибору підходящої платформи для хостингу. У випадку використання платформи Telegram, необхідно забезпечити, щоб інфраструктура хостингу була здатна підтримувати високу доступність та безпеку, оскільки бот буде постійно взаємодіяти з користувачами через цей месенджер. При цьому важливо врахувати вимоги Telegram до ботів та їх API для інтеграції.

Після вибору платформи для хостингу наступним кроком є налаштування систем моніторингу та логування. Це дозволяє відслідковувати стан бота в

реальному часі та швидко реагувати на будь-які проблеми, що можуть виникнути. Моніторинг має охоплювати різні аспекти роботи бота, включаючи його продуктивність, час відгуку та точність виявлення фішингу. Логування ж важливе для забезпечення збору даних про взаємодію бота з користувачами та його роботу в цілому, що може бути корисним для подальшого аналізу та вдосконалення системи.

Нарешті, автоматизація процесів розгортання є ключовою для забезпечення швидкого та безпомилкового запуску бота в експлуатацію. Це включає скрипти та інструменти для автоматичного розгортання оновлень, що дозволяє мінімізувати простої та помилки, які можуть виникнути внаслідок ручного втручання. Автоматизація також важлива для підтримки безперервної інтеграції та доставки (CI/CD)[39], що є сучасним підходом до розробки та експлуатації програмного забезпечення.

Загалом, процес розгортання та моніторингу бота вимагає ретельного планування та впровадження низки технічних рішень, які забезпечать його стабільну та ефективну роботу у середовищі Telegram.

2.8 Оцінка ефективності та прийняття рішень

Основними інструментами оцінки є збір та аналіз статистики використання, що включає в себе кількісні та якісні показники роботи системи[40]. Це може бути частота виявлення фішингових повідомлень, точність класифікації електронних листів, швидкість реагування бота та взаємодія користувачів з системою після отримання попереджень.

Далі, ми використовуємо зібрані дані для аналізу[41], який допомагає ідентифікувати сильні сторони бота та області, які потребують удосконалення. Цей аналіз є критичним для планування наступних ітерацій розвитку, дозволяючи команді розробників встановлювати пріоритети в удосконаленні функціоналу та оптимізації алгоритмів.

Завдяки цим методикам оцінки, команда забезпечує постійний цикл зворотного зв'язку, який є необхідним для адаптації бота до змінних умов та

нових викликів у сфері кібербезпеки. В результаті, ці методики формують основу для прийняття обґрунтованих рішень щодо подальшого розвитку та впровадження нових функцій у бота.

2.9 Висновки до розділу 2

У другому розділі було розроблено теоретичну основу для створення системи виявлення фішингових повідомлень на базі Telegram-бота. Було детально описано архітектуру системи, визначено основні функціональні блоки та розроблено алгоритми для аналізу вмісту повідомлень. Особливу увагу було приділено розробці зручного інтерфейсу взаємодії користувача з ботом, що передбачає інтуїтивно зрозумілі команди та повідомлення. Для аналізу вмісту повідомлень буде здійснено детальний аналіз посилань, що містяться в тексті, а також перевірка їхньої легітимності за допомогою спеціалізованих баз даних. Паралельно буде проведено пошук типових схем фішингу та шаблонних фраз, характерних для таких повідомлень. Отримані результати дозволяють сформувати чітке уявлення про структуру та принципи роботи майбутньої системи, що є необхідним кроком для її подальшої практичної реалізації.

РОЗДІЛ 3
РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ВИЯВЛЕННЯ ФІШИНГОВИХ АБО ШАХРАЙСЬКИХ ПОВІДОМЛЕНЬ У ЕЛЕКТРОННІЙ ПОШТІ КОРИСТУВАЧА

3.1 Вибір інструментів для розробки

Для реалізації даного застосунку було вибрано ось такі мови програмування та бібліотеки до них:

Python: Ця мова програмування є популярним вибором для розробки Telegram ботів завдяки своїй простоті, гнучкості та широкому набору бібліотек. Вона має просту синтаксичну структуру, що робить її легкою для вивчення та використання, навіть для початківців. Python має безліч бібліотек, які можуть бути корисними для розробки бота, таких як Telethon, NLTK, scikit-learn та інші.

Бібліотеки та фреймворки:

Telethon [42]: Ця бібліотека Python дозволяє створювати Telegram ботів з широким набором функцій. Вона надає доступ до всіх можливостей Telegram API, що дозволяє боту спілкуватися з користувачами, надсилати повідомлення, обробляти команди та багато іншого.

NLTK [43]: Ця бібліотека Python пропонує інструменти для обробки природної мови, які можуть використовуватися для аналізу тексту електронних листів та виявлення ознак фішингу. Вона може допомогти боту розуміти значення тексту, визначати ключові слова, класифікувати емоції та багато іншого.

scikit-learn [44]: Ця бібліотека Python пропонує алгоритми машинного навчання, які можуть використовуватися для класифікації електронних листів як фішингових або легітимних. Вона може допомогти боту автоматично розпізнавати шкідливі повідомлення.

Для забезпечення ефективної роботи та підтримки системи було використано ряд інструментів та фреймворків.

3.1.1 СУБД MySQL

MySQL є однією з найпопулярніших систем управління реляційними базами даних (СУБД) у світі[45]. Вона використовується для зберігання, організації та управління даними в багатьох веб-додатках та корпоративних системах. MySQL є відкритим програмним забезпеченням, що забезпечує високу продуктивність, надійність та масштабованість.

Основні характеристики MySQL:

1) **Відкритий код.** MySQL є проєктом з відкритим кодом, що означає, що він доступний для безкоштовного використання та модифікації. Це робить його доступним для широкого кола користувачів та розробників.
2) **Підтримка реляційної моделі даних.** MySQL використовує реляційну модель даних, що дозволяє зберігати дані в таблицях, які можуть бути пов'язані між собою за допомогою ключів. Це забезпечує ефективне зберігання та доступ до даних.
3) **Висока продуктивність.** MySQL оптимізований для швидкої обробки запитів та високої продуктивності, що робить його ідеальним вибором для веб-додатків та систем з великими обсягами даних.
4) **Масштабованість.** MySQL може бути використаний як для невеликих додатків, так і для великих корпоративних систем з великою кількістю користувачів та великими обсягами даних. Він підтримує масштабування як вертикальне (додавання ресурсів до одного сервера), так і горизонтальне (додавання нових серверів).
5) **Безпека.** MySQL забезпечує високий рівень безпеки даних за допомогою таких функцій, як аутентифікація користувачів, шифрування даних та контроль доступу на основі ролей. Це дозволяє захистити дані від несанкціонованого доступу.
6) **Підтримка транзакцій.** MySQL підтримує транзакції, що дозволяє забезпечити цілісність даних та виконання складних операцій з даними. Це особливо важливо для додатків, де важлива точність та надійність даних.

7) **Широка підтримка та спільнота.** MySQL має велику спільноту користувачів та розробників, що забезпечує доступ до великої кількості ресурсів, документації та підтримки. Це робить його легким у вивченні та використанні.
8) **Інтеграція з іншими технологіями.** MySQL легко інтегрується з різними мовами програмування (PHP, Python, Java, C++, і т.д.) та іншими технологіями (веб-сервери, інтерфейси API, системи управління контентом). Це робить його універсальним інструментом для розробки різноманітних додатків.

MySQL є надійним та ефективним інструментом для управління базами даних, що забезпечує високий рівень продуктивності, безпеки та масштабованості. Його використання дозволяє створювати надійні та ефективні інформаційні системи для виявлення фішингових або шахрайських електронних листів, забезпечуючи зберігання та обробку великих обсягів даних.

3.1.2 Flask

Flask – це мікрофреймворк для розробки веб-додатків на Python, створений Арміном Ронахером, як частина проєкту Pocoo. Він був випущений у 2010 році і швидко здобув популярність завдяки своїй простоті, гнучкості та можливості швидкого розгортання веб-додатків. На відміну від більш важких фреймворків, таких як Django, Flask пропонує мінімалістичний підхід, що дозволяє розробникам зосередитися на написанні коду без необхідності дотримуватися суворих правил та структур.

Основні переваги Flask[46]:

- **Легкість та простота.** Flask має мінімалістичний дизайн, що дозволяє розробникам швидко розпочати роботу без необхідності розбиратися в складних налаштуваннях. Це робить його ідеальним для невеликих проєктів та прототипів.
- **Гнучкість.** Flask не має жорстких обмежень щодо структури проекту, що дозволяє розробникам організовувати код так, як їм зручно. Це

забезпечує велику свободу у виборі архітектури додатку.

- **Розширюваність.** Flask підтримує розширення через різні плагіни та бібліотеки, що дозволяє додавати нові функціональні можливості. Існує безліч розширень для роботи з базами даних, аутентифікації користувачів, обробки форм та багато іншого.
- **Сумісність.** Flask легко інтегрується з іншими бібліотеками та інструментами Python, такими як SQLAlchemy для роботи з базами даних або Jinja2 для шаблонізації. Це дозволяє розробникам використовувати знайомі інструменти та технології.
- **Спільнота та документація.** Flask має велику та активну спільноту, що забезпечує підтримку та розвиток фреймворку. Документація Flask є детальною та доступною, що полегшує процес навчання та розробки.

Основні компоненти Flask-додатку

Для створення функціонального веб-додатку за допомогою Flask необхідно розуміти його основні будівельні блоки. Кожен компонент Flask-додатку відіграє важливу роль у процесі обробки запитів і генерації відповідей. Далі ми розглянемо детально кожен з них.

- Flask app object (Об'єкт додатку Flask): Це основний об'єкт, який представляє ваш веб-додаток. Він відповідає за налаштування маршрутизації, обробку запитів та інші важливі функції.
- Маршрутизація (Routing): Flask використовує декоратори для визначення маршрутів, що дозволяє легко вказувати, які функції обробляють певні URL. Наприклад, для створення маршруту, який обробляє запити до кореневого URL, використовується наступний код:
- Шаблони (Templates): Flask використовує Jinja2 як шаблонізатор, що дозволяє легко створювати динамічні HTML-сторінки. Шаблони дозволяють відокремити логіку додатку від його представлення, що полегшує підтримку та розвиток коду.
- Конфігурація (Configuration): Flask дозволяє легко налаштовувати

додаток через конфігураційні файли або змінні середовища. Конфігураційні параметри можуть бути завантажені з файлу config.py або встановлені безпосередньо у коді.

- Розширення (Extensions): Flask має безліч розширень, які додають додаткові можливості, такі як робота з базами даних, аутентифікація користувачів, обробка форм та багато іншого. Наприклад, для інтеграції з базою даних можна використовувати розширення Flask-SQLAlchemy.

Flask дозволяє розробникам швидко створювати веб-додатки, забезпечуючи при цьому гнучкість та розширюваність. Завдяки своїй простоті та можливості інтеграції з іншими інструментами, Flask став популярним вибором серед розробників Python для створення веб-додатків різної складності. Його мінімалістичний підхід дозволяє розробникам зосередитися на написанні коду, а не на налаштуваннях фреймворку, що робить його ідеальним вибором для багатьох проектів.

3.1.3 Gmail API

Gmail API – це програмний інтерфейс, який надає розробникам доступ до функціональності електронної пошти Gmail. За допомогою цього API можна взаємодіяти з електронною поштою користувачів, виконуючи різноманітні операції, такі як читання, надсилання, видалення та пошук повідомлень, а також робота з мітками.

Gmail API використовує HTTP-запити для взаємодії з серверами Google. Кожен запит до API виконується за допомогою певного HTTP-методу (GET, POST, DELETE тощо) і містить необхідні параметри та дані. API відповідає на ці запити, повертаючи результати у форматі JSON, що дозволяє легко обробляти їх у додатку.

Gmail API використовується для інтеграції функціональності Gmail у додатки та сервіси. Це може бути корисним у різних сценаріях, таких як:

- Автоматизація обробки електронної пошти: створення скриптів для автоматичного сортування, фільтрації або відповіді на повідомлення.

- Інтеграція з іншими сервісами: об'єднання електронної пошти з іншими бізнес-додатками, такими як CRM-системи або інструменти для управління завданнями.
- Аналіз електронної пошти: витягування даних з повідомлень для подальшого аналізу, створення звітів або візуалізацій.

Переваги використання Gmail API[47]

- Доступ до повного функціоналу Gmail: API надає можливість використовувати всі основні функції Gmail, такі як читання, надсилання та видалення повідомлень, робота з мітками та інше.
- Безпека та надійність: використання OAuth 2.0 для автентифікації та авторизації забезпечує безпечний доступ до даних користувача.
- Гнучкість та масштабованість: API дозволяє створювати додатки, які можуть обробляти велику кількість повідомлень та користувачів, забезпечуючи високу продуктивність та ефективність.
- Інтеграція з іншими Google API: можливість комбінувати функціональність Gmail API з іншими API від Google, такими як Google Calendar API або Google Drive API, для створення комплексних рішень.

Використання Gmail API відкриває широкі можливості для розробників, дозволяючи інтегрувати функціональність електронної пошти у свої додатки та сервіси, автоматизувати рутинні завдання та створювати нові інноваційні рішення.

3.1.4 SafeBrowsing API

SafeBrowsing API[48] – це програмний інтерфейс, розроблений компанією Google, який забезпечує доступ до бази даних небезпечних веб-сайтів. Цей API дозволяє розробникам перевіряти URL-адреси на наявність фішингових, шахрайських або шкідливих сайтів, що допомагає захистити користувачів від інтернет-загроз.

SafeBrowsing API працює шляхом перевірки URL-адрес проти бази даних небезпечних сайтів, яку постійно оновлює Google. Коли додаток надсилає запит

до API з певною URL-адресою, API відповідає, вказуючи, чи знаходиться ця адреса в списку небезпечних. Відповідь може включати інформацію про тип загрози, наприклад, фішинг, шкідливе програмне забезпечення або небажане програмне забезпечення.

SafeBrowsing API використовується для підвищення безпеки інтернет-користувачів, забезпечуючи захист від фішингових і шахрайських сайтів, а також від сайтів, що розповсюджують шкідливе програмне забезпечення.

Основні сценарії використання включають:

- Захист веб-браузерів: інтеграція API у веб-браузери для перевірки кожної відвідуваної сторінки на наявність загроз.
- Безпека електронної пошти: перевірка URL-адрес у повідомленнях електронної пошти на наявність фішингових посилань.
- Захист мобільних додатків: використання API для перевірки посилань, відвідуваних користувачами мобільних додатків, на безпеку.
- Фільтрація контенту: забезпечення безпеки користувачів при доступі до веб-контенту через різні онлайн-сервіси.

Переваги використання SafeBrowsing API[49]

- Висока точність та актуальність: база даних небезпечних сайтів постійно оновлюється Google, що забезпечує високу точність виявлення загроз.
- Швидка інтеграція: API легко інтегрується у різні додатки та сервіси, забезпечуючи швидке впровадження захисту.
- Масштабованість: API здатний обробляти великий обсяг запитів, що дозволяє використовувати його у великих проектах з великою кількістю користувачів.
- Універсальність: SafeBrowsing API може бути використаний у різних контекстах, від веб-браузерів до мобільних додатків та електронної пошти, що робить його універсальним інструментом для забезпечення безпеки.

Використання SafeBrowsing API допомагає захистити користувачів від

інтернет-загроз, підвищуючи загальний рівень безпеки при роботі в інтернеті. Інтеграція цього API у вашу інформаційну систему для виявлення фішингових або шахрайських повідомлень у електронній пошті користувача забезпечить надійний захист від небезпечних посилань, що є ключовим елементом у боротьбі з інтернет-шахрайством.

3.1.5 Telegram Bot API

Telegram Bot API – це програмний інтерфейс, розроблений для створення та керування ботами у месенджері Telegram[50]. Боти – це спеціальні акаунти, які не вимагають телефонного номера для реєстрації і можуть виконувати автоматизовані дії, взаємодіючи з користувачами через текстові повідомлення, команди та інтерактивні елементи.

Telegram Bot API працює за принципом обміну HTTP-запитами між сервером розробника та серверами Telegram[51]. Основні функції API включають:

- Надсилання та отримання повідомлень: боти можуть надсилати текстові повідомлення, зображення, відео, документи та інші типи контенту, а також отримувати повідомлення від користувачів.
- Обробка команд: боти можуть реагувати на спеціальні команди, які користувачі надсилають у чаті, наприклад, /start, /help тощо.
- Інтерактивні елементи: боти можуть використовувати кнопки, інлайн-клавіатури та інші інтерактивні елементи для покращення взаємодії з користувачами.
- Вебхуки: боти можуть налаштовувати вебхуки для отримання повідомлень у реальному часі, що дозволяє миттєво реагувати на дії користувачів.

Telegram Bot API використовується для створення автоматизованих систем, які можуть виконувати різноманітні завдання, взаємодіючи з користувачами через Telegram. Основні сценарії використання включають:

- Автоматизація підтримки клієнтів: боти можуть відповідати на

запитання користувачів, надавати інформацію та допомагати вирішувати проблеми.

- Інформаційні сервіси: боти можуть надсилати користувачам новини, оновлення, сповіщення та іншу важливу інформацію.
- Розважальні сервіси: боти можуть надавати розважальний контент, такі як ігри, вікторини, анекдоти тощо.
- Інтеграція з іншими сервісами: боти можуть взаємодіяти з іншими системами та сервісами, наприклад, для бронювання, замовлення послуг, отримання даних тощо.

Переваги використання Telegram Bot API[52]

- Простота використання: Telegram Bot API має добре документований інтерфейс, що дозволяє розробникам швидко створювати та налаштовувати ботів.
- Гнучкість та масштабованість: API дозволяє створювати боти з різними функціями та масштабувати їх під потреби проекту.
- Миттєва взаємодія: завдяки підтримці вебхуків боти можуть миттєво реагувати на дії користувачів, забезпечуючи високу швидкість обробки запитів.
- Інтерактивні можливості: використання кнопок, інлайн-клавіатур та інших інтерактивних елементів покращує взаємодію з користувачами та робить її більш зручною.
- Безпека: Telegram забезпечує високий рівень безпеки, шифруючи повідомлення між ботом та користувачем.

Використання Telegram Bot API у інформаційній системі для виявлення фішингових або шахрайських повідомлень у електронній пошті користувача дозволить створити зручний та ефективний інструмент для взаємодії з користувачами, надаючи їм можливість швидко отримувати інформацію та допомогу через популярний месенджер.

3.2 Розробка елементів системи

3.2.1 База даних

База даних є центральним компонентом системи, який зберігає всі необхідні дані для обробки та аналізу повідомлень. Основні таблиці бази даних можуть включати:

Ключі доступу: інформація про користувачів системи (telegram_id, last message,credentials).

Повідомлення: збережені електронні листи для аналізу (ID повідомлення, telegram_id користувача, тема повідомлення, дата отримання , інформація про відправника).

3.2.2 API інтеграції

API допомагають інтегрувати різні системи та додатки, забезпечуючи їхню сумісність і розширюваність[53]. Тому у проекті було використано декілька API:

1. Gmail API
    - Забезпечує доступ до поштових скриньок користувачів для отримання повідомлень.
    - Використовується для отримання нових повідомлень та їх збереження у базі даних для подальшого аналізу.
2. SafeBrowsing API
    - Використовується для перевірки URL-адрес у повідомленнях на наявність фішингових або шахрайських сайтів.
    - Повертає результати перевірки, які використовуються для оцінки безпеки повідомлень.
3. Telegram Bot API
    - Використовується для створення та керування Telegram ботом, який взаємодіє з користувачами системи.
    - Забезпечує надсилання повідомлень користувачам, отримання команд та запитів, а також надання результатів аналізу.

3.2.3 Логіка обробки та аналізу повідомлень

Логіка обробки та аналізу повідомлень включає кілька етапів[54]:

1. Отримання повідомлень: за допомогою Gmail API система отримує нові повідомлення користувачів(після генерації токена доступу, функція з деякою періодичністю буде перевіряти наявність нових повідомлень за допомогою змінної lastmessage, яка зберігає дату останнього перевіреного повідомлення, і якщо такі будуть надсилати на перевірку).
2. Аналіз вмісту[55]: система аналізує вміст повідомлень, виявляючи підозрілі елементи, такі як фішингові URL-адреси короткі домени, підозрілі слова, посилання обгортки тощо.
3. Перевірка URL-адрес: за допомогою SafeBrowsing API система перевіряє URL-адреси у повідомленнях на наявність фішингових або шахрайських сайтів.
4. Збереження результатів: результати обробляються у форматі відсотка, який показує наскільки ймовірно повідомлення фішингове, і повертаються у телеграм бот.
5. Надсилання сповіщень: через Telegram бот користувачі отримують результати аналізу та додаткову інформацію.

3.3 Структура бази даних

База даних складається з трьох таблиць, одна зберігає дані про користувачів, друга інформацію про всі листи, а третя інформацію про перевірки. Зв’язані вони по ключам user_id для швидкого повернення результатів у telegram користувача.

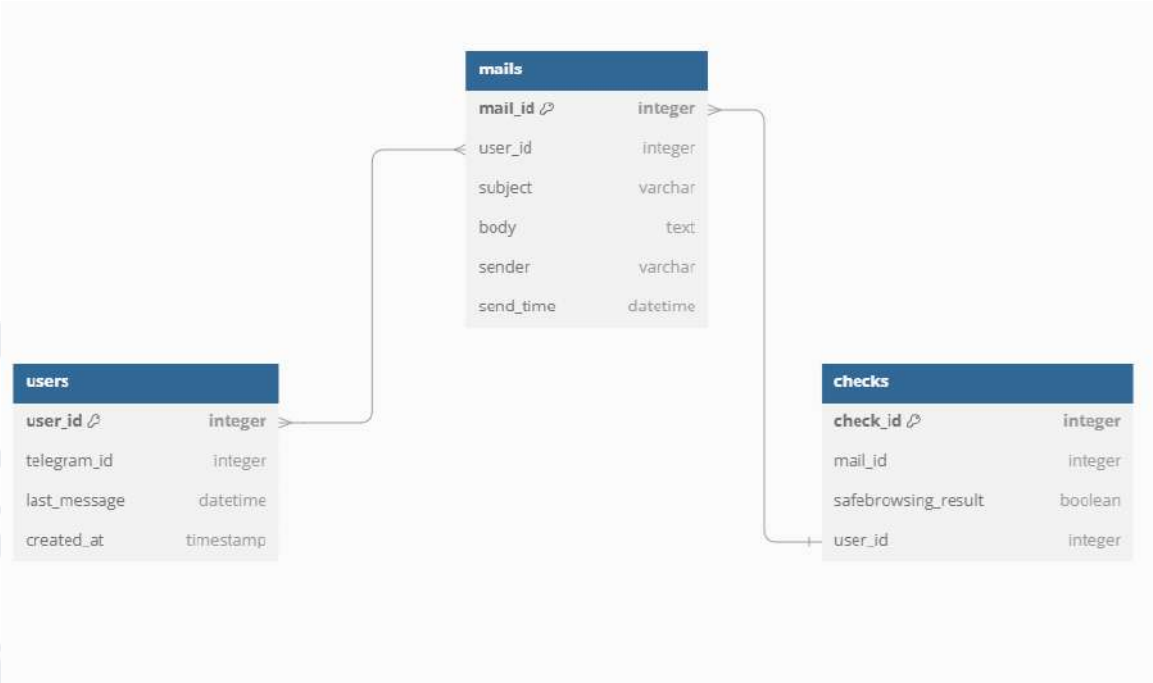


Рис 3.1. Фрагмент бази даних

Опис основних таблиць та їх зв'язків

У даній системі використовуються три таблиці, одна з них зберігає інформацію про електронні листи. Таблиця Mails має наступну структуру:

Таблиця 3.1 – Тип та опис полів таблиці Mails

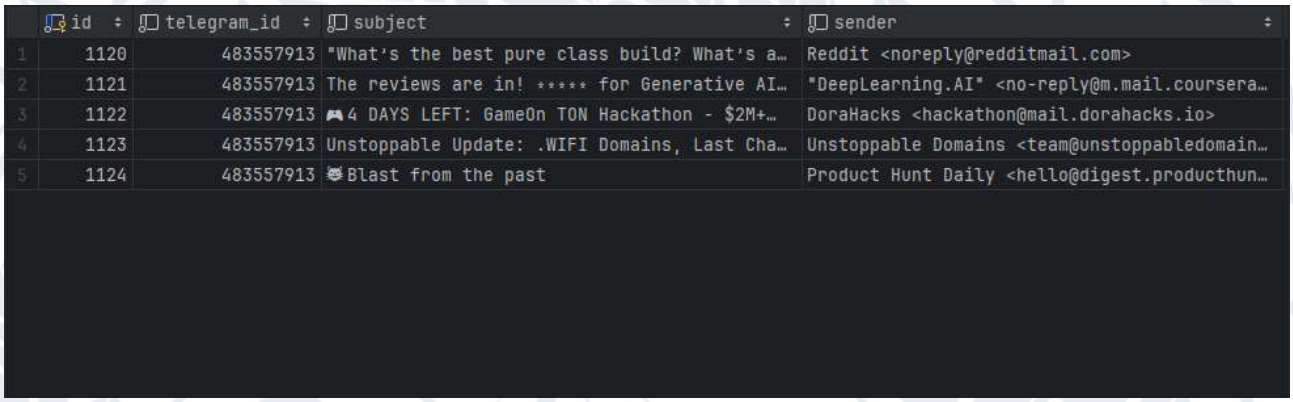
Назва	Тип	Опис
mail_id	UUID	Унікальний ідентифікатор
user_id	INT	Ідентифікатор юзера у телеграмі
subject	Varchar(256)	Тема
sender	Varchar(256)	Відправник
body	Varchar(256)	Тіло листа
send_time	datetime	Час відправки

mail_id: це поле є первинним ключем таблиці та автоматично збільшується для кожного нового запису.

user_id: це поле зберігає ідентифікатор користувача. Воно використовується для

зв'язку електронного листа з конкретним користувачем. А також з його токенами доступу, які зберігаються в іншій таблиці.

Subject, body, sender, send_time: ці поля потрібні для подальшого інформування користувача при перевірці на фішинг. Після перевірки вказується час відправлення, відправник та тема листа, щоб його простіше було ідентифікувати.



	id	telegram_id	subject	sender
1	1120	483557913	"What's the best pure class build? What's a…	Reddit <noreply@redditmail.com>
2	1121	483557913	The reviews are in! ***** for Generative AI…	"DeepLearning.AI" <no-reply@m.mail.coursera…
3	1122	483557913	🎮 4 DAYS LEFT: GameOn TON Hackathon - $2M+…	DoraHacks <hackathon@mail.dorahacks.io>
4	1123	483557913	Unstoppable Update: .WIFI Domains, Last Cha…	Unstoppable Domains <team@unstoppabledomain…
5	1124	483557913	🐞 Blast from the past	Product Hunt Daily <hello@digest.producthun…

Рис. 3.2. Приклад заповнення таблиці Mails

Інша таблиця users, яку було створено для зберігання інформації про користувачів Telegram та їх токени доступу до Gmail API. Це дозволяє зв'язати кожен Telegram ID з відповідним токеном доступу, що необхідно для інтеграції з Gmail API.

Таблиця users [i]містить наступні поля:

- user_id: Унікальний ідентифікатор користувач
- telegram_id: Ідентифікатор користувача у Telegram.
- last_message Дата та час останнього повідомлення від користувача.
- Created_at: Час створення користувача.

Остання таблиця checks призначена для зберігання результатів перевірки електронних листів на наявність ознак фішингу або інших загроз. Ці результати дозволяють відстежувати історію перевірок, аналізувати ефективність методів перевірки та надавати користувачам детальну інформацію про потенційні ризики.

Таблиця 3.2 – Тип та опис полів таблиці users.

Назва	Тип	Опис
user_id	UUID	Ідентифікатор користувача
telegram_id	INT	Ідентифікатор юзера у телеграмі
last_message	DATETIME	Дата повідомлення
Created_at	DATETIME	Токен Доступу

	telegram_id	last_message	credentials
1	480656426	2024-11-14 14:04:41	{"token": "ya29.a0AeDClZBP_PNQHeLzLM8x3HYk0E
2	483557913	2024-11-14 19:42:05	{"token": "ya29.a0AeDClZAlQV69cHC7SMmDCDtOWF
3	552096713	2021-11-15 08:29:05	<null>
4	5933565247	2024-11-14 16:17:24	<null>

Рис. 3.3. Приклад заповнення таблиці users

Структура таблиці:

Оскільки ми не маємо конкретних деталей щодо методів перевірки, які використовуються в системі, наведемо загальний шаблон структури таблиці checks, який можна адаптувати під ваші конкретні потреби:

Таблиця 3.3 – Тип та опис полів таблиці Checks

Назва	Тип	Опис
check_id	UUID	Унікальний ідентифікатор перевірки
mail_id	INT	Ідентифікатор листа
Safebrowsing_result	BOOLEAN	Результат перевірки посилань через API SafeBrowsing
User_id	INT	Ідентифікатор користувача

3.4 Реалізація функціоналу

3.4.1 Реєстрація користувачів та зберігання листів

Опис процесу реєстрації та зберігання листів

Реєстрація користувачів відбувається через бота у Telegram. Процес включає наступні етапи:

1) Команда старт: Користувач починає взаємодію з ботом, вводячи команду “/start”. Бот відповідає повідомленням з інструкціями та видає посилання для автентифікації. У телеграмі це виглядає так:

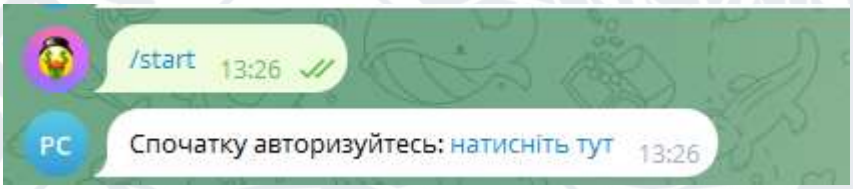


Рис 3.4. Вигляд у боті

2) Перехід за посиланням: Користувач переходить за наданим посиланням, яке перенаправляє його на сторінку автентифікації Google. На цій сторінці користувач підключає свою електронну пошту та підтверджує надання доступу до читання листів.

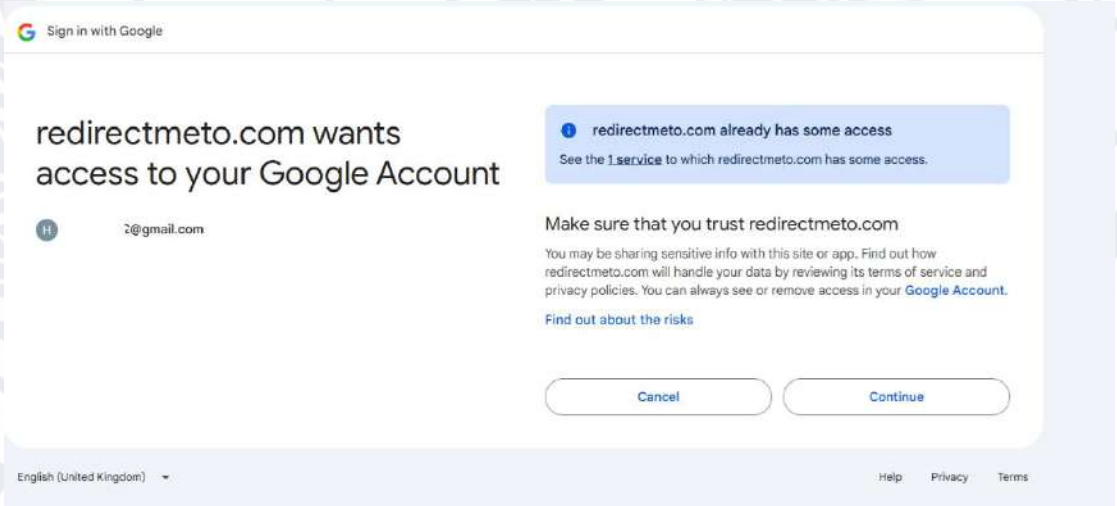


Рис 3.5. Вигляд вікна переходу

3) Отримання токену: Після успішної автентифікації, система отримує OAuth токен, який дозволяє доступ до електронної пошти

користувача.

4) Запис даних у базу даних: Токен з Telegram ID користувача зберігаються у базі даних MySQL. Це забезпечує можливість ідентифікації користувача та доступу до електронної пошти.
5) Завантаження останніх листів: Система автоматично отримує останні 10 електронних листів з поштової скриньки користувача через Gmail API. Ці листи зберігаються у базі даних для подальшого аналізу.

Цей процес забезпечує безпечну та зручну реєстрацію користувачів, а також автоматичне завантаження листів та їх подальший аналіз.

3.4.2 Процес перевірки листів

Опис процесу перевірки листів

У програмі є функція some_periodic_function(), яка періодично перевіряє чи є нові листи серед усіх користувачів і якщо такі наявні, запускає для них перевірку.

3.4.3 Функція перевірки листів is_dangerous()

Мета функції:

Функція is_dangerous призначена для аналізу тексту електронного листа з метою визначення ймовірності того, що цей лист є фішинговим. Алгоритм функції базується на пошуку характерних ознак фішингових листів, таких як:

Ключові слова: наявність слів і словосполучень, які часто використовуються в фішингових листах (наприклад, "терміново", "перевірити", "акаунт", "пароль" тощо).

Посилання: аналіз URL-адрес, що містяться в листі, на предмет їхньої безпеки та наявності ознак, характерних для фішингових сайтів (наприклад, коротких URL, підозрілих доменів).

Принцип роботи:

1. Виділення ключових слів: Функція проходить по тексту листа і шукає збіги зі списком ключових слів. Чим більше збігів, тим вища ймовірність того, що лист є фішинговим.

2. Аналіз URL-адрес:

- Перевірка в кеші безпечних URL: Спочатку функція перевіряє, чи не міститься URL-адреса в кеші безпечних URL. Якщо такий запис знайдено, то URL вважається безпечним.
- Виклик API Google Safe Browsing: Якщо URL не знайдено в кеші, функція звертається до API Google Safe Browsing для перевірки URL на наявність загроз. Генерація запита обробляється в окремому методі check_url_safe_browsing. За допомогою токена, генерується запит з посиланням і передається по API. У результаті отримуємо чи знаходиться посилання в реєстрі фішингових посилань
- Аналіз характеристик URL: Для URL, які не визнано безпечними, функція аналізує такі характеристики:
    - Довжина домену: короткі домени можуть бути підозрілими.
    - Наявність спеціальних символів: символи "@", "%", "xn--" можуть свідчити про спробу обманути користувача.
    - Короткі URL: використання сервісів скорочення URL може приховувати справжню адресу сторінки.
    - Підозрілі домени: домени, що складаються з однієї частини або мають IP-адресу замість доменного імені, можуть бути підозрілими.
- Розрахунок балу ризику**:** На основі кількості знайдених ключових слів та кількості підозрілих URL-адрес функція розраховує бал ризику. Чим вищий бал, тим більша ймовірність того, що лист є фішинговим.

Використання кешу:

Для підвищення швидкості роботи функція використовує кеш безпечних URL. Це дозволяє уникнути повторних звернень до API Google Safe Browsing для URL, які вже були перевірені раніше.

Важливі моменти:

- Список ключових слів: Список ключових слів, які використовуються функцією, може бути розширений і уточнений в міру появи нових схем фішингу.
- Порог ризику: Для прийняття рішення про те, чи є лист фішинговим, необхідно встановити поріг ризику. Листи, які набрали бал ризику вище за поріг, вважаються підозрілими. Порог потрібно постійно уточнювати в межу вдосконалення функції.
- Помилково позитивні та негативні результати: Незважаючи на високу точність, функція може давати помилкові позитивні (безпечні листи класифікуються як фішингові) та помилкові негативні (фішингові листи класифікуються як безпечні) результати. Для зменшення кількості помилок необхідно постійно вдосконалювати алгоритм і список ключових слів.

Функцію можна розширити за рахунок використання інших методів аналізу тексту, таких як машинне навчання, для підвищення точності виявлення фішингових листів.

3.4.4 Взаємодія з користувачем через Telegram

Використання Telegram Bot API для сповіщень

Для взаємодії з користувачами та надсилання сповіщень використовується Telegram Bot API[56]. Цей API дозволяє боту надсилати повідомлення користувачам, інформуючи їх про потенційно небезпечні електронні листи та інші важливі події.

Формат повідомлень для користувачів

Повідомлення, які надсилаються користувачам через Telegram, мають наступний формат:

*Message '{subject}' from '{sender}' at {timestamp.isoformat()} is potentially dangerous ({confidence * 100}% confidence)!*

У такому форматі користувач чітко бачить усе необхідне, а саме:

- Інформація про відправника: ім’я та пошта з якої був відправлений лист.
- Час отримання: Точний час, коли лист було отримано, у форматі ISO.
- Рівень небезпеки: Відсоток впевненості в тому, що лист є потенційно небезпечним.

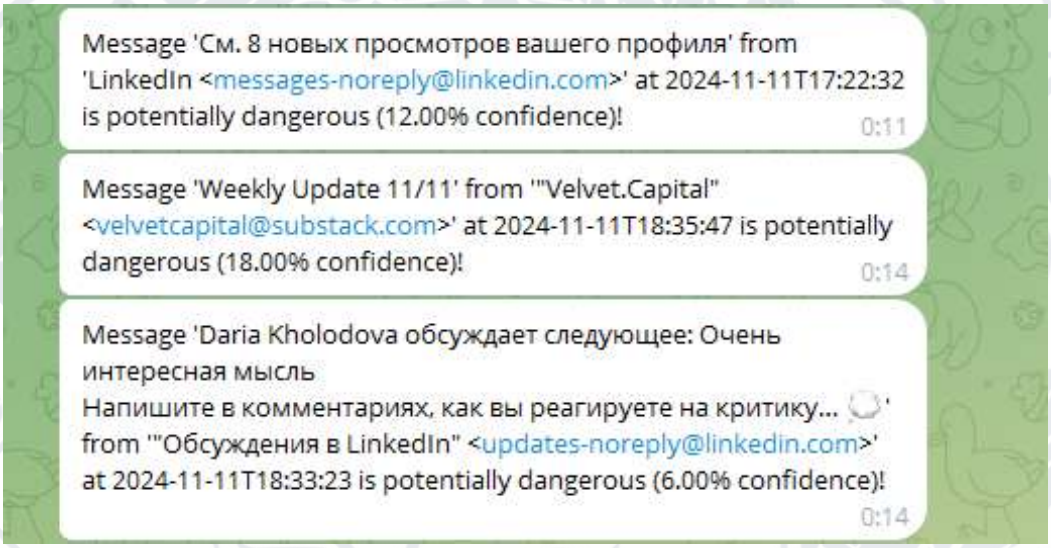


Рис 3.6. Приклад проаналізованих повідомлень

3.5 Оцінка результатів

Для оцінки точності та ефективності системи[57] аналізу повідомлень було проведено тестування на реальних даних. Основні результати тестування показують, що система успішно ідентифікує потенційно небезпечні повідомлення.

Система використовує рівень впевненості у небезпеці для класифікації листів. На основі отриманих результатів можна зробити такі висновки:

- **Повідомлення з рівнем небезпеки до 15%**: Ці листи мають низький рівень ризику і їх можна не перевіряти додатково. Вони вважаються безпечними для відкриття і не потребують особливої уваги.

- **Повідомлення з рівнем небезпеки від 15% до 40%**: Ці листи мають середній рівень ризику і їх варто перевіряти з обережністю. Рекомендується звертати увагу на деталі таких листів і бути обачливими при взаємодії з ними.
- **Повідомлення з рівнем небезпеки понад 40%**: Ці листи мають високий рівень ризику і їх слід повністю ігнорувати та не відкривати. Вони вважаються потенційно небезпечними і можуть містити фішингові або шахрайські елементи.

Таким чином, система демонструє високу ефективність[58] у виявленні потенційно небезпечних листів та забезпечує користувачам своєчасні сповіщення про можливі загрози. Це сприяє підвищенню рівня кібербезпеки користувачів, дозволяючи їм зосередитися на більш ризикованих листах і уникати потенційних загроз.

3.6 Висновки до розділу 3

У цьому розділі було детально розглянуто процес розробки та функціонування системи для виявлення фішингових або шахрайських повідомлень у користувацьких електронних листах. Система базується на інтеграції декількох ключових компонентів, кожен з яких відіграє важливу роль у забезпеченні її ефективності та надійності.

Використані технології: для зберігання даних обрано MySQL, а для розробки веб-додатку – Flask. Це забезпечило надійність і гнучкість системи. Інтеграція з Gmail API дозволяє системі аналізувати вхідні листи користувачів, а SafeBrowsing API допомагає виявляти шкідливі URL. Для взаємодії з користувачами використовується Telegram Bot API.

Основна функціональність системи базується на функції is_dangerous(), яка аналізує текст листа, шукає підозрілі слова та перевіряє URL-адреси. На основі цього аналізу визначається рівень ризику листа.

Система повідомляє користувачів про потенційно небезпечні листи через Telegram, надаючи детальну інформацію про лист. Тестування показало високу

точність роботи системи. Загалом, розроблена система ефективно виявляє фішингові атаки, захищаючи користувачів від шахрайства.

Ключові переваги системи:

- Висока точність виявлення фішингових листів
- Зручний інтерфейс взаємодії з користувачем
- Інтеграція з популярними сервісами (Gmail, Telegram)
- Можливість налаштування та розширення функціоналу

Перспективи розвитку:

- Розширення функціоналу за рахунок аналізу вкладених файлів та інших типів даних.
- Покращення точності класифікації за допомогою нових алгоритмів машинного навчання.
- Розробка мобільного додатку для зручнішого використання системи.

Розроблена система є цінним інструментом для боротьби з фішингом і може бути використана як окремий продукт або як компонент більших систем кібербезпеки.

ВИСНОВКИ

В ході виконання курсової роботи була докладно вивчена сучасна мережа фішингових взаємодій. Розібрано усі типи фішингу та їх загрози. Також було вивчено безліч сервісів-аналогів, які допомагають захистити себе у інтернеті. Також було розібрано, як створювати телеграм-бота та як використовувати відому фішингову базу для зменшення шансів попадання фішингу на пошту.

Основним етапом у вивченні створення телеграм бота було розібрати, як працюють аналоги та вивчити їх переваги та недоліки, щоб використати це при побудові свого бота. Було розібрано як використовувати gmail API для отримання доступу до змісту пошти.

Практика показала, що телеграм бот є дуже зручним інструментом і може бути використаний у багатьох напрямах, адже популярність соцмереж росте, тому інструменти, які побудовані всередині соцмереж будуть і надалі ставати більш популярними і поширюватись серед користувачів.
В рамках подальшого дослідження можна розглянути такі теми:

- Покращення точності виявлення фішингових повідомлень за допомогою більш досконалих методів машинного навчання Дослідження таких методів, як глибоке навчання, нейронні мережі та трансформери, може бути частиною цього.
- Створити методи, які можуть автоматично блокувати фішингові веб-сайти. Це може включати вивчення таких методів, як чорні списки URL-адрес, аналіз поведінки користувачів і блокування доступу до веб-сайтів, які є шкідливими.
- Підключення інформаційної системи до інших систем безпеки. Інтеграція з антивірусними програмами, системами захисту від вторгнень і системами запобігання втратам даних є можливими варіантами.

СПИСОК ВИКОРИСТАНИХ ПОСИЛАНЬ

1. Oles, N. How to Catch a Phish: A Practical Guide to Detecting Phishing Emails. , 2021, 356 с.
2. Що таке фішинг та як захистити від нього свої особисті дані URL: https://mc.today/uk/shho-take-fishing/ (дата звертання: 25.05.2024)
3. Headline Phishing Statistics URL: https://aag-it.com/the-latest-phishing-statistics (дата звертання: 25.05.2024)
4. Dove, M. The Psychology of Fraud, Persuasion and Scam Techniques: Understanding What Makes Us Vulnerable, 2022, (1st ed., 435 pp.).
5. Kevin Mitnick "Social Engineering: The Art of Human Hacking", 2021, 533 с.
6. Kevin Mitnick and Robert C. Davis "The Art of Deception: Controlling the Human Element of Security", 2002, 416 с.
7. How can I tell if an email is phishing? URL: https://servicehub.ucdavis.edu/servicehub?id=ucd_kb_article&sysparm_article=KB0008380 (дата звертання: 25.05.2024)
8. Маннинг К.Д., Рагхаван П., Шютце Х. " Введення в інформаційний пошук", 2011, 512с.
9. El-Sayed El-Metwaly, A., Bedair, M. R., ... Takieldeen, A. E. (2024). Detection of Phishing URLs Based on Machine Learning and Cybersecurity. 2024 International Telecommunications Conference (ITC-Egypt).
10. Kaspersky site URL: https://ukraine.kaspersky.com/ (дата звертання: 25.05.2024)
11. Norton 360 site URL: https://safeweb.norton.com/ (дата звертання: 25.05.2024)
12. Phishing Secure extension https://chromewebstore.google.com/detail/threatslayer-security-phi/mgcmocglffknmbhhfjihifeldhghihpj?hl=uk (дата звертання: 25.05.2024)

13. Proofpoint Email Protection https://www.proofpoint.com/sites/default/files/2020-04/pfpt-us-ds-email-protection.pdf (дата звертання: 25.05.2024)
14. NoScript Extension https://chromewebstore.google.com/detail/noscript/doojmbjmlfjjnbmnoijecmcbfeoakpjm?hl=uk (дата звертання: 25.05.2024)
15. The Telegram phishing market. URL: https://securelist.com/telegram-phishing-services/109383/ (дата звертання: 25.05.2024)
16. Devising and Detecting Phishing Emails Using Large Language Models URL:https://www.researchgate.net/publication/378883394_Devising_and_detecting_phishing_emails_using_large_language_models (дата звертання: 25.05.2024)
17. A User's Guide: 10 Ways to Protect Your Personal Data URL: https://www.infosecinstitute.com/resources/security-awareness/a-users-guide-10-ways-to-protect-your-personal-data/ (дата звертання: 25.05.2024)
18. Gmail API documentation URL: https://developers.google.com/gmail/api/guides (дата звертання: 25.05.2024)
19. SafeBrowsing API documentation URL: https://developers.google.com/safe-browsing/v4 (дата звертання: 25.05.2024)
20. "Telegram Bot API" URL: https://core.telegram.org/ . (дата звертання: 25.05.2024).
21. Ma, J., Zhang, Y., Li, Y., & Wang, S. (2016). A survey on phishing attacks and defense mechanisms. IEEE Communications Surveys & Tutorials, 18(3), 1686-1706.
22. Create and Host a Telegram Bot URL: https://codecapsules.io/tutorial/how-to-create-and-host-a-telegram-bot-on-code-capsules/ (дата звертання: 25.05.2024).
23. Telegram Bot Features URL: https://core.telegram.org/bots/features (дата звертання: 25.05.2024).

24. Telegram bot security solutions URL: https://medium.com/@bioogrami7/telegram-bot-security-solutions-6c5b8105a0e1 (дата звертання: 25.05.2024)
25. Scaling Up II: High Load URL: https://grammy.dev/advanced/scaling (дата звертання: 25.05.2024)
26. Telegram Bot Logger URL: https://pypi.org/project/telegram-bot-logger/ (дата звертання: 25.05.2024)
27. Telegram Bot Interface URL: https://smarthomeconnect.readthedocs.io/en/latest/interfaces/telegram.html (дата звертання: 25.05.2024)
28. MySQL and Telegram integration URL: https://n8n.io/integrations/mysql/and/telegram/ (дата звертання: 25.05.2024)
29. Connect Gmail and Telegram Bot integrations URL: https://www.make.com/en/integrations/google-email/telegram (дата звертання: 25.05.2024)
30. Phishing Detection in Depth URL: https://zvelo.com/phishing-detection-in-depth/ (дата звертання: 25.05.2024)
31. Проста логістична регресія URL: https://ukrayinska.libretexts.org/ (дата звертання: 25.05.2024)
32. Ансамблі моделей машинного навчання URL: https://evergreens.com.ua/ua/articles/ensembles.html (дата звертання: 25.05.2024)
33. Telegram OAuth Authorization for Your Site URL: https://dev.to/shaggyrec/telegram-oauth-authorization-for-your-site-3f4l (дата звертання: 25.05.2024)
34. Setup Telegram bot to get alert notifications URL: https://medium.com/linux-shots/setup-telegram-bot-to-get-alert-notifications-90be7da4444 (дата звертання: 25.05.2024)

35. Telegram Bot — The New User Interface URL: https://medium.com/the-hooray-media/telegram-bot-the-new-user-interface-e96047615109 (дата звертання: 25.05.2024)
36. Cybersecurity Telegram channels, groups, bots URL: https://telegramchannels.me/tag/cybersecurity (дата звертання: 25.05.2024)
37. How to Test a Telegram Bot URL: https://medium.com/@duketemon/how-to-test-a-telegram-bot-ba54eb1cadc0 (дата звертання: 25.05.2024)
38. Enhancing User Engagement with Multiselection Inline Keyboards in Telegram Bots URL: https://medium.com/@moraneus/enhancing-user-engagement-with-multiselection-inline-keyboards-in-telegram-bots-7cea9a371b8d (дата звертання: 25.05.2024)
39. Що таке CI/CD URL: https://corewin.ua/blog/ci-cd-devsecops/ (дата звертання: 25.05.2024)
40. A high-accuracy phishing website detection method based on machine learning URL: https://www.sciencedirect.com/science/article/abs/pii/S2214212623001370 (дата звертання: 25.05.2024)
41. Analytical Method Validation: are your analytical methods suitable for intended use? URL: https://qbdgroup.com/en/blog/analytical-method-validation-are-your-methods-suitable-for-intended-use/ (дата звертання: 25.05.2024)
42. Telethon documentation URL: https://docs.telethon.dev/en/stable/ (дата звертання: 25.05.2024)
43. NLTK documentation URL: https://www.nltk.org/book/ (дата звертання: 25.05.2024)
44. scikit-learn documentation URL: https://scikit-learn.org/stable/index.html (дата звертання: 25.05.2024)
45. Why SQL Is the Perfect Database Language URL: https://learnsql.com/blog/database-language/ (дата звертання: 25.05.2024)

46. Why Should You Use Flask for Web Development? URL: https://www.planeks.net/why-use-flask/ (дата звертання: 25.05.2024)
47. Gmail API – why you should consider using it URL: https://mailtrap.io/blog/send-emails-with-gmail-api/ (дата звертання: 25.05.2024)
48. Likarish, P., & Jung, E. (2009). Leveraging Google SafeBrowsing to characterize web-based attacks. Computer Science.
49. Kurniawan, Y., Connie, C., Nirwana, N. (2023). The development media for interactive learning using bots API of Telegram social media on harmonic motion materials for class X of senior high school. Kasuari Physics Education Journal (KPEJ).
50. Telegram Bot API Integrations URL: https://pipedream.com/apps/telegram-bot-api (дата звертання: 25.05.2024)
51. Automating Telegram Bot Deployment with GitHub Actions and Docker URL: https://tjtanjin.medium.com/automating-telegram-bot-deployment-with-github-actions-and-docker-482abcd2533e (дата звертання: 25.05.2024)
52. ANALYSIS OF CONNECTION METHODS OF TELEGRAM ROBOTS WITH SERVER PART URL: https://journals.nmetau.edu.ua/index.php/st/article/view/73/21 (дата звертання: 25.05.2024)
53. Sabir, B., Babar, M., & Gaire, R. (2020). An evasion attack against ML-based phishing URL detectors. arXiv.org.
54. Zhao, J., Li, Y., & Liu, S. (2019). Deep learning for phishing detection: A survey. IEEE Access, 7, 56440-56457.
55. Chari, G., Sheffer, B., & D'ippolito Asapp, N. (2023). Scaling web API integrations. 2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP).
56. Rahayu, Y. S., Wibawa, S. C., Kusumadewi, S. (2018). The development of BOT API social media. IOP Conference Series: Materials Science and Engineering.

57. Kim, J. R., Shim, W., Yoon, H., Hong, S. H., Lee, J. S., Cho, Y., & Kim, S. (2017). Evaluation of the Accuracy and Efficiency. AJR. American journal of roentgenology.
58. Begum J, N. (2024). An automated daily news reports generating application involving keyword-based news scraping, summarization, and sentimental analysis leveraging NLP models. International Journal for Research in Applied Science and Engineering Technology.

ДЕКЛАРАЦІЯ

про дотримання академічної доброчесності

Я, Лаврентюк Назар Юрійович, освітньої програми комп'ютерні технології обробки даних (Data Science), інформаційна система для виявлення фішингових або шахрайських повідомлень у електронній пошті користувача, що нижче підписався, розуміючи та підтримуючи загальновизнані засади справедливості, доброчесності та законності,

ЗОБОВ'ЯЗУЮСЬ:

дотримуватися принципів та правил академічної доброчесності, що визначені законодавством України, локальними нормативними актами Донецького національного університету імені Василя Стуса, положеннями, правилами, умовами, визначеними іншими суб'єктами, та не допускати їх порушення.

ПІДТВЕРДЖУЮ:

що мені відомі положення статті 42 Закону України «Про освіту»;

що у даній роботі не представляла/представляв чиїсь роботи повністю або частково як свої власні. Там, де я скористалася/скористався працею інших, я зробила/зробив відповідні посилання на джерела інформації;

що дана робота не передавалась іншим особам і подається вперше, не порушує авторських та суміжних прав закріплених статтями 21-25 Закону України «Про авторське право та суміжні права», а дані та інформація не отримувались в недозволений спосіб.

УСВІДОМЛЮЮ:

що ця робота може бути перевірена університетом на плагіат або інші порушення академічної доброчесності, в тому числі з використанням спеціалізованих сервісів;

що у разі порушення академічної доброчесності, до мене можуть бути застосовані процедури, передбачені законодавством України та Кодексом академічної доброчесності та корпоративної етики Донецького національного університету імені Василя Стуса, іншими локальними нормативними актами університету, та я можу бути притягнута/притягнутий до академічної відповідальності.

_____01.12.2025___ _____

(дата) (підпис)

ALMA MATER
NOMEN EST OMEN
ІМ'Я ЗОБОВ'ЯЗУЄ