Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

Система реєстрації та публікації інцидентів кібербезпеки з використанням технології блокчейн.

Текстова частина до курсової роботи
за спеціальністю «Комп'ютерні науки» - 122

Керівник курсової роботи
Старший викладач
Гороховський К.С.
_____ (Підпис)
"___" _____ 2024 року
Виконав студент
КН-4 Волков І. О.
"___" _____ 2024 року

Київ 2024

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Старший викладач

_____ Гороховський К.С.

„_____” _____ 2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Волкову Івану Олексійовичу

факультету інформатики 4 курсу бакалаврської програми

ТЕМА: **Система реєстрації та публікації інцидентів кібербезпеки з використанням технології блокчейн.**

Зміст ТЧ до курсової роботи:
Індивідуальне завдання
Вступ
Розділ 1. Дослідження та аналіз предметної області
Розділ 2. Розділ 2. Аналіз технічного завдання
Розділ 3. Розробка застосунку
Висновки
Список літератури
Додатки (за необхідністю)

Дата видачі „___” _____ 2024 р.

Керівник_____

(підпис)

Завдання отримав_____

(підпис)

Календарний план виконання роботи

№	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	01.10.2023	
2.	Огляд літератури за темою роботи	01.11.2023	
3.	Проведення дослідження	01.12.2023	
4.	Написання програмного застосунку	01.01.2024	
5.	Написання текстової частини	04.04.2024	
6.	Захист курсової роботи	20.05.2024	

Студент _____

Керівник _____ “_____”_____ 2024

Зміст

Перелік умовних позначень:

Enisa - The European Union Agency for Cybersecurity.

IPFS - InterPlanetary File System.

CIRAS - Cybersecurity Incident Reporting and Analysis System.

Анотація

Анотація:

Ця робота присвячена розробці та реалізації системи реєстрації та публікації інцидентів кібербезпеки з використанням технології блокчейн. В контексті постійно зростаючих загроз кібербезпеці, ефективне виявлення, відстеження та аналіз інцидентів стає критично важливим завданням для забезпечення безпеки та надійності інформаційних систем.

Використання технології блокчейн в даній системі обрано з метою забезпечення недоторканності даних про інциденти кібербезпеки. Блокчейн, як децентралізована технологія зберігання даних, надає надійний механізм запису та підтвердження інформації, що робить його ідеальним вибором для створення системи реєстрації інцидентів кібербезпеки. Децентралізований характер блокчейну також забезпечує стійкість до змін та відмов, а також високу ступінь захищеності від втручання з боку небажаних сторін.

Ця робота буде детально розглядати процес проектування та реалізації застосунку на дану тему, включаючи вибір технологічних рішень, аналіз вимог до безпеки та протоколів забезпечення конфіденційності, а також ефективність та масштабованість розробленої системи.

Ключові слова: блокчейн, смарт контракт, IPFS, децентралізація, кібербезпека

Вступ

Актуальність теми

Вступ:

З кожним роком світ стає все більш цифровим, що призводить до зростання кількості кіберінцидентів та загроз кібербезпеці. Якщо брати до уваги період з 2012 по 2023 роки, то згідно з статистикою Enisa, загальна кількість кібер інцидентів на території ЄС збільшилась у 10 разів:

Рис 1.1 - статистика по кількості кібер інцидентів від Enisa

Ця тенденція вимагає негайних заходів для ефективного виявлення, реагування та подолання наслідків кібератак. Підприємства та організації з усього світу зазнають значних втрат через кіберінциденти, які можуть посягати на їхню фінансову стійкість, репутацію та навіть безпеку клієнтів.

З цієї причини стає дедалі важливіше розвивати та удосконалювати системи реєстрації та публікації інцидентів кібербезпеки. Ефективне рапортування кібер інцидентів від компаній до державних регуляторів відіграє критичну роль у протидії кіберзагрозам та відновленні нормального функціонування після атаки. Однак існуючі системи рапортування часто мають обмежену ефективність та можуть страждати від недоліків у форматі, безпеці та достовірності даних.

У даному дослідженні ми досліджуємо можливість використання технології блокчейн для створення системи реєстрації та публікації

інцидентів кібербезпеки, що має потенціал покращити ефективність та безпеку процесу рапортування в основному в контексті ЄС, але тенденція шириться також і на ринок США, на момент написання роботи (2024 рік) комісія з цінних паперів і бірж (SEC) наказала зареєстрованим компаніям, включаючи криптокомпанії, публікувати щорічні звіти про «управління ризиками кібербезпеки, стратегію та управління».

Нове правило вимагає від компаній розкривати будь-які «суттєві» інциденти кібербезпеки протягом чотирьох робочих днів у спробі поглибити довіру між інвесторами та публічними компаніями. Компанії повинні детально вказати, як кібератака вплине на їхній бізнес, разом із звітом із детальним описом інциденту та часу.

Отже вивчення цієї теми є надзвичайно актуальним у зв'язку з постійним зростанням кількості кібер інцидентів та потребою у вдосконаленні заходів захисту та реагування на них.

Об’єкт дослідження

Приватність, незмінність, безпечність та зручність подання звітів про кібер інциденти.

Предмет дослідження

Децентралізована модель подання звітів з використанням IPFS (InterPlanetary File System) для забезпечення децентралізованого сховища даних.

Мета дослідження

Перевірити та підтвердити ефективність підходу до обраного формату подання звітів.

Постановка задачі

1. Аналіз структури та шаблонів існуючих звітів європейських компаній за минулі роки , а також вимог європейських регуляторів.
2. Запровадження отриманої структури до смарт-контракту.
3. Дослідження ефективної моделі аутентифікації та авторизації користувачі(сторін) системи.
4. Розробка фронт-енд застосунку з використанням отриманих даних, використовуючи структури даних смарт-контракту.

Структура роботи:

1) В першому розділі викладено загальні відомості про предметну область, описано наявні проблеми та запропоновано, та обґрунтовано підхід до їх вирішення. Крім цього, проаналізовано існуючі рішення на ринку.
2) В другому розділі проаналізовано технічне завдання: описано користувачів, функціональні вимоги та вимоги до даних.
3) В третьому розділі обґрунтовано вибір інструментів, детально описано реалізацію ключових компонентів системи та наведено результати роботи програми.

Розділ 1. Дослідження та аналіз предметної області

1.1 Загальні відомості, приклади/статистика інцидентів від Enisa (The European Union Agency for Cybersecurity).

У 2018 році набула чинності директива ЄС [1] про безпеку мережевих та інформаційних систем (так звана Директива NIS), яка запровадила правила сповіщення про інциденти кібербезпеки для операторів основних послуг у багатьох критичних секторах, таких як енергетика, транспорт, фінанси та охорона здоров'я. У ЄС постачальники критично важливих послуг повинні повідомляти про інциденти кібербезпеки зі значним впливом національні органи влади у своїй країні. Наприкінці кожного року зведені звіти про ці інциденти збираються, знеособлюються, узагальнюються та аналізуються ENISA.

Якщо порівняти два найбільші за кількістю кібер атак роки (2022-2023), то можемо побачити що зловмисна природа інциденту , а тобто кібератака всього за рік збільшилась на 5%, отже очевидно що у майбутньому варто чекати продовження даної тенденції. Та навіть якщо і не так, то банальні помилки людей та системні помилки також можуть вплинути на роботу системи, отже про них теж треба звітувати.

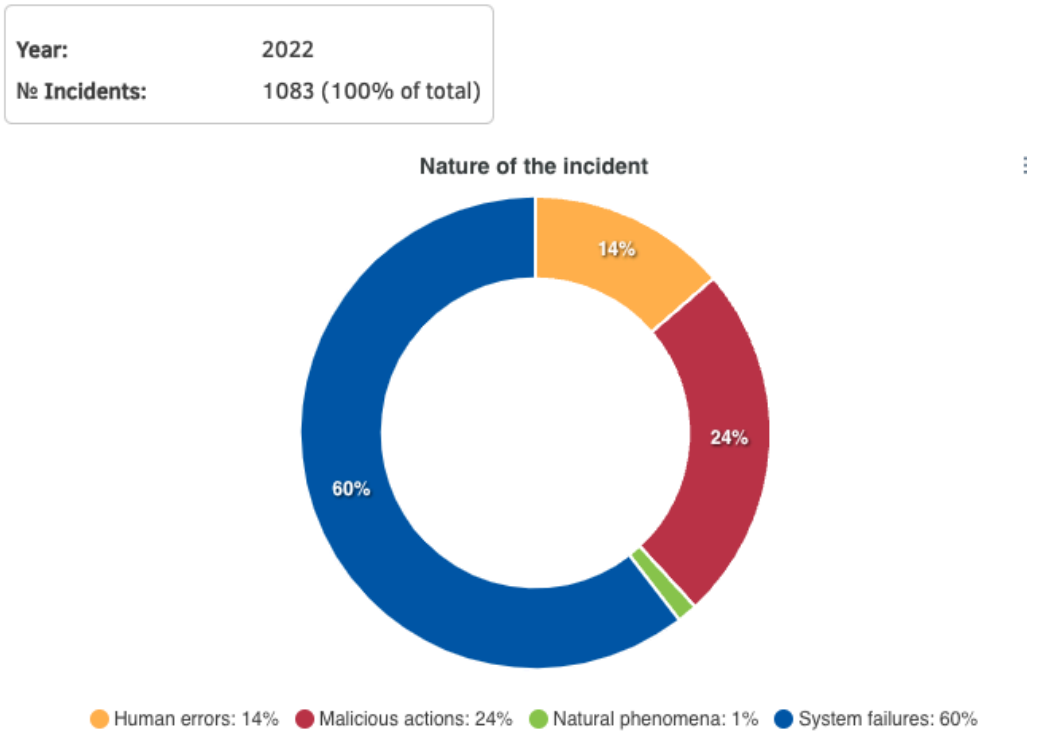


Рис 1.2 - природа кібер інцидентів за 2022 рік

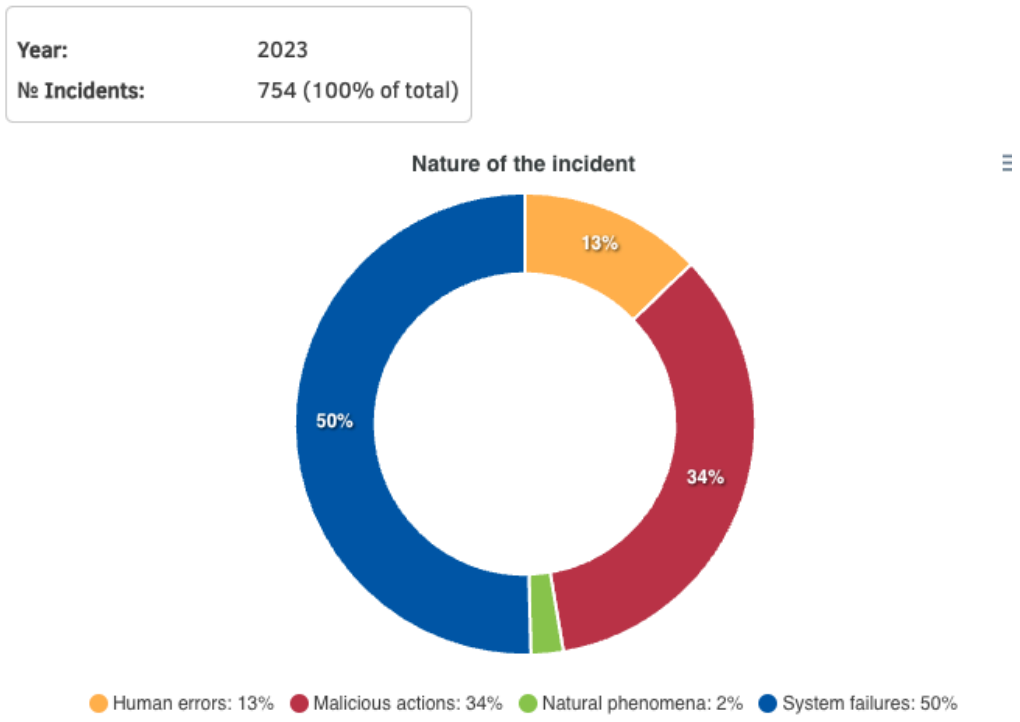


Рис 1.3 - природа кібер інцидентів за 2023 рік

Тенденція ширяться також і на ринок США, на момент написання роботи (2024 рік) комісія з цінних паперів і бірж (SEC) наказала зареєстрованим компаніям, включаючи криптокомпанії, публікувати щорічні звіти про «управління ризиками кібербезпеки, стратегію та управління».Нове правило[3] вимагає від компаній розкривати будь-які «суттєві» інциденти кібербезпеки протягом чотирьох робочих днів у спробі поглибити довіру між інвесторами та публічними компаніями.

1.2 Проблематика

За статистикою найбільше кібер атак відбувається в таких сферах як communications та health.

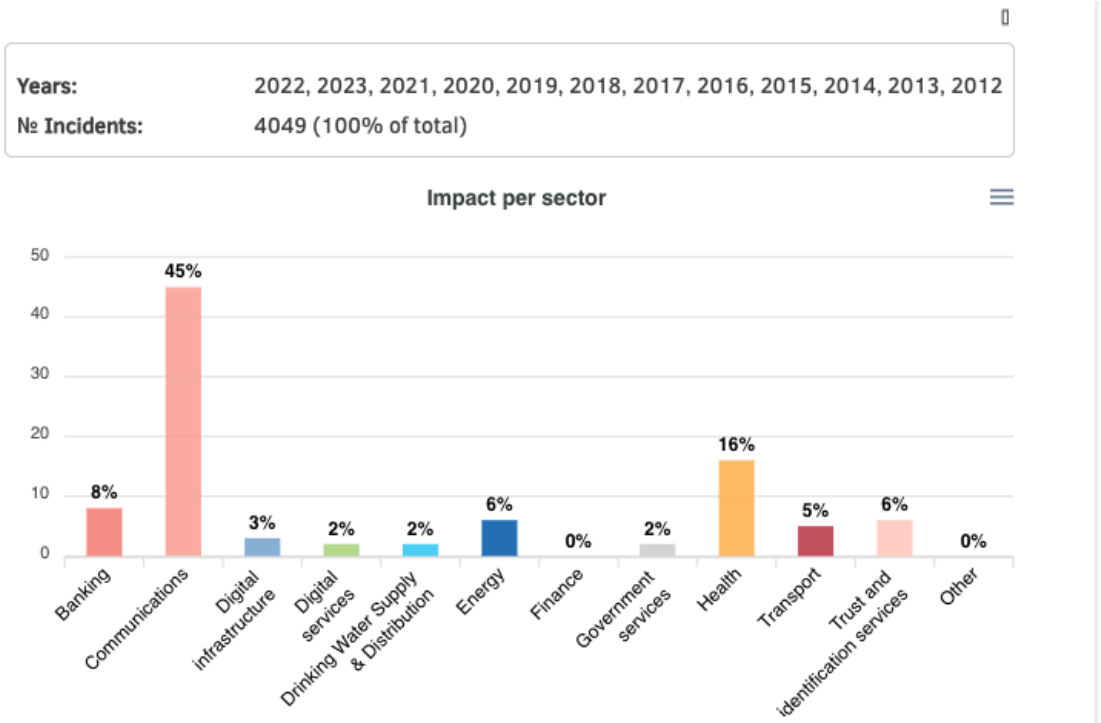


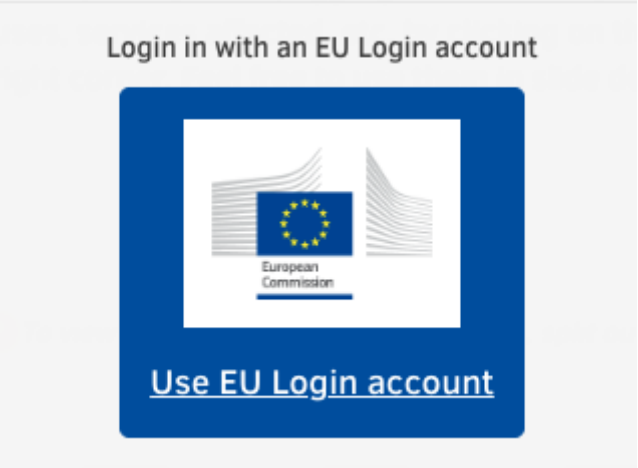
Рис 1.4 - вплив кібер атак на сектори

Спеціалісти з департаменту промислової та системної інженерії Об'єднаних Арабських Еміратів робили дослідження[2] щодо ефективності публікації звітів у сфері здоров'я, вони зазначають, що

недостатнє звітування - велика проблема. Сучасні централізовані органи які надають послуги звітування як правило розробляються державними компаніями і мають багато незручностей, серед них : незручний інтерфейс, потреба створення облікового запису та регулярної аутентифікації, можливість втрати даних через людський фактор або кібер атаку, відсутність оптимізованого зворотнього зв’язку, що в результаті призводить до втрати мотивації або страху у людей до подання звітів, їх некоректність або прострочення термінів звітування. Щоб усунути згадані вище обмеження в поточних системах звітності, технологія блокчейн може надати можливості завдяки своїм унікальним функціям, такі як прозорість, незмінність, конфіденційність тощо. Одна з ключових ідей системи звіту - зручний та ефективний перегляд історії звітів про інциденти, у роботі лікарень зокрема, для можливості швидко доступитися до даних у разі повторення спроб кібер атак, тому цей аспект також буде опрацьовано в застосунку.

1.3 Аналіз наявних рішень поставленої задачі

ENISA підтримує CIRAS, систему звітування та аналізу інцидентів кібербезпеки. Щоби перевірити її функціонал достатньо створити аккаунт (EU Login Account).



Сайт повільний та в принципі не дав мені змоги протестувати його функціонал. Таке буває якщо продукт розробляє не приватна а державна компанія, тим не менш, вони надають зручні шаблони для заповнення звітів, що однозначно полегшує роботу компаніям. Також існують системи які вже інтегрували в застосунок блокчейн, проте це скоріше системи заточені на конкретній сфері(наприкод Telecom), метою ж цієї роботи є створення універсальної платформи для подання звітів, яку можна було б використовувати в різних сферах.

1.4 Опис обраного підходу до вирішення поставленої задачі

За основу буде обрано децентралізований підхід до зберігання даних та прав доступу до них за допомогою блокчейну, смарт-контракту, та децентралізованого сховища файлів. В цілому ми даних підходом позбуваємось проблем звичайних баз даних таких як централізація, і як наслідок можливість втрати всіх даних внаслідок вразливості.

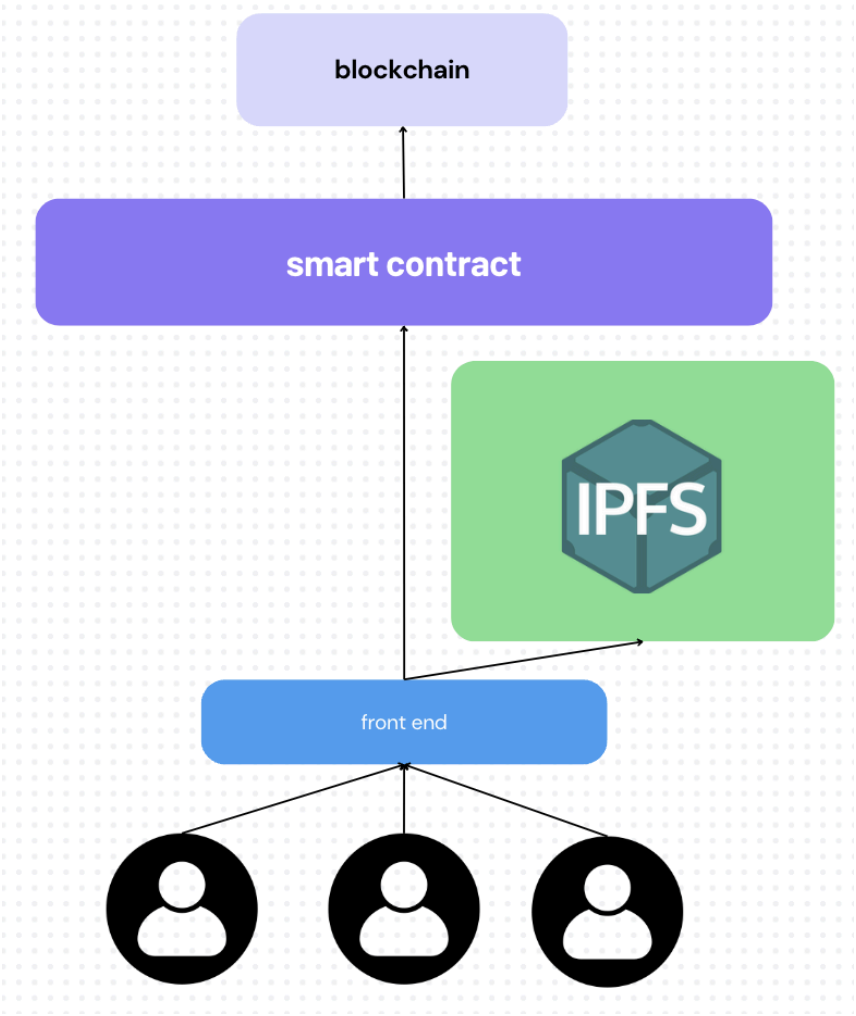


Рис 1.5 - ідея вирішення проблеми

Проте звичайну базу даних також можна і варто використовувати в таких застосунках задля збереження швидкості взаємодії користувача з застосунком, в контексті програми її можна використовувати для збереження метаданих про звіти, для безкоштовного та швидкого отримання в разі якщо пропускна здатність застосунку мала.

1.5 Обґрунтування вибору рішення

Blockchain має кілька функцій, які можуть допомогти у вирішенні проблем, з якими стикаються поточні системи звітності про інциденти. Поточні прогалини можуть підсумовуватися таким чином: відсутність поширення інформації в режимі реального часу, відсутність стимулу, відсутність безпеки та конфіденційності, фрагментація даних про несприятливі події між різними організаціями та неможливість мати конструктивний зворотний зв’язок щодо того, чи повідомлення про інцидент призвело до її вирішення. Фактично, ключові аспекти блокчейну, такі як послідовність, безпека та конфіденційність даних, децентралізація, прозорість, стимули та можливість відстеження, можуть бути корисними для забезпечення покращення систем звітності. По суті, мінімізуючи кількість централізованих органів зберігання та маніпулювання даними, ми робимо ставку на надійність та довготривалість роботи застосунку. Проте як уже згадувалось блокчейн має як переваги так і недоліки.

З недоліків виділяють :

- Масштабованість. Мережевий трафік блокчейну стає громіздким, оскільки кількість транзакцій збільшується щодня. Кожен вузол у блокчейні повинен зберігати всі перевірені транзакції, і це стає перешкодою, оскільки існує обмеження на розмір блоку та часовий інтервал, який використовується для створення нового блоку.
- Юридичні виклики. До цієї дати смарт-контракти та блокчейн загалом сильно нерегульовані та нестандартизовані на національному та міжнародному рівнях.

Table 1 Comparison between Using a Centralized Database and a Blockchain Platform

Feature	Traditional Centralized Database	Blockchain Platform
Authority	Controlled by a central authority (administrator)	Authority is shared among stakeholders and is decentralized
Data Integrity	Data can be altered	Data is auditable and immutable
Data handling	Can support only four primary operations: read, create, update, and delete	Only read and write options are available
Data Privacy	High chances of malicious cyber-attacks	Transaction data is stored in blocks using cryptography technology
Data Provenance	Databases cannot ensure that data has not been altered, forged, reproduced, or stolen	Users can trace and verify the provenance of all the previous transactions by accessing any node in the network
Transparency	Databases are not as transparent as in BC	Transaction data is stored in a distributed network
Quality assurance	Administrators are needed to authenticate data	Data can be traced from its origin using cryptography technology
Fault tolerance	Considerable threat of single point of failure (SPF)	The ledger is fault-tolerant
Incentive	There is no incentives mechanism in databases	Health professionals and patients can be incentivized for reporting promptly and accurately
Consensus	Databases do not have a consensus mechanism as they are centralized	The validation mechanism ensures the integrity of the data as much as possible
Cost	Easy implementation and maintenance as it is a conventional technology	Limited certainty in operation and maintenance costs
Performance	Fast (more transactions processed per second) and offer high scalability	Can handle minimal transactions per second and has as scalability since the technology is at its developing phase

Рис 1.6 - [2] порівняння блокчейну та звичайної бази даних

1.5.1 Обґрунтування вибору блокчейну Optimism

Блокчейн Optimism - це масштабована розширена версія блокчейну Ethereum, яка працює на основі технології Optimistic Rollups. Основна особливість Optimistic Rollups полягає в тому, що вони використовують оптимістичне підтвердження транзакцій, що дозволяє використовувати значно менше обчислювальних ресурсів порівняно з традиційними методами підтвердження транзакцій.

Переваги використання блокчейну Optimism включають:

1. Масштабованість: завдяки використанню технології Optimistic Rollups, блокчейн Optimism може обробляти значно більше транзакцій за один час порівняно з традиційним блокчейном Ethereum.

2. Зниження вартості транзакцій: оптимістичне підтвердження транзакцій дозволяє зменшити витрати на операції з обчисленнями, що забезпечує більш доступні та ефективні транзакції.

3. Збереження сумісності: блокчейн Optimism зберігає сумісність з Ethereum, що означає, що додатки та розробники можуть безперешкодно переходити з Ethereum на Optimism та навпаки, не втрачаючи сумісності зі сховищами даних та іншими ресурсами.

Розділ 2. Аналіз технічного завдання

Головна мета застосунку

Забезпечити прозорий, децентралізований, надійний та зручний спосіб зберігання звітів від користувачів та їх надійна передача іншим користувачам.

2.1 Опис користувачів в системі

Якщо компанія дуже велика і має багато департаменів то в такому випадку доцільна модель “Employee Wallet <> Company Wallet <> Regulator Wallet” , де у кожного члена організації може бути свій гаманець , він може надсилати звіт на гаманець компанії, а представники компанії вже в свою чергу надсилають звіти на гаманець регулятора.

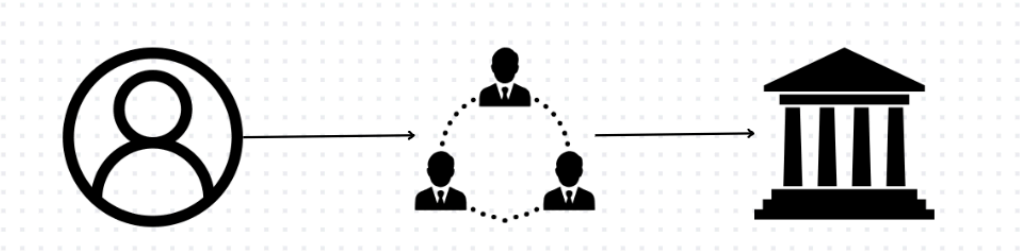


Рис 2.1 - структура користувачі в системі

Якщо невелика, тоді логічно використовувати модель “Company Wallet <> Regulator Wallet”.

2.2 Технічні ролі користувачів у системі

В контексті однієї взаємодії ролі виглядають наступним чином :

Звітуючий:

- завантажувати звіти
- передивлятися звіти
- надавати/забирати доступ до звітів

Регулятор:

- завантажувати звіти
- передивлятися звіти звітуючих
- фільтрувати звіти
- надавати зворотній зв’язок

Розділ 3. Розробка застосунку

3.1 Обґрунтування вибору засобів розробки

3.1.1 Solidity

Для написання смарт-контрактів, які використовуються на блокчейні, потрібно використовувати мову програмування, яка може взаємодіяти з певною блокчейн-платформою і реалізовувати логіку контрактів. Solidity є однією з таких мов, і вона спеціально призначена для написання смарт-контрактів на блокчейні Ethereum.

Основні особливості Solidity включають:

1. Синтаксис, схожий на мову програмування JavaScript, що робить її зрозумілою для багатьох розробників.
2. Можливість використання об'єктно-орієнтованого програмування для створення складних смарт-контрактів.
3. Вбудовані функції для взаємодії з блокчейном Ethereum, такі як робота з транзакціями та доступ до даних.
4. Можливість використання умов для автоматизації різноманітних операцій та логіки контракту.
5. Підтримка бібліотек та інші можливості для полегшення розробки складних смарт-контрактів.
6. Зручні структури даних, які легко використовувати в контексті даної задачі.

3.1.2 Hardhat та Alchemy

Hardhat - це набір інструментів для розробки смарт-контрактів на блокчейні Ethereum. Він надає розробникам зручний та потужний набір інструментів для розробки, тестування та розгортання смарт-контрактів. За допомогою Hardhat розробники можуть швидко і ефективно створювати та випробовувати свої смарт-контракти, використовуючи сучасні практики розробки програмного забезпечення.

Щодо Alchemy, це інфраструктурний провайдер для розробників децентралізованих додатків (DApps) на блокчейні Ethereum. Він надає доступ до надійних, швидких та масштабованих вузлів Ethereum API. За допомогою Alchemy розробники можуть легко взаємодіяти з мережею Ethereum, відправляти транзакції, читати дані з ланцюжка блоків та інші операції. Alchemy допомагає забезпечити стабільність та швидкодію взаємодії з Ethereum, що робить його корисним інструментом для розробників смарт-контрактів.

3.1.3 IPFS та Pinata

IPFS (InterPlanetary File System) - це протокол для розподіленого зберігання та обміну даними[4]. Використання IPFS при розробці застосунку для зберігання звітів про кіберінциденти буде мати наступні переваги:

1. Децентралізоване зберігання: IPFS дозволяє зберігати дані на різних вузлах мережі, замість централізованого сервера. Це забезпечує

більшу стійкість до відмов та атак, оскільки дані розподілені по всій мережі.

2. Висока доступність: завдяки децентралізованому характеру, дані, збережені за допомогою IPFS, можуть бути доступні навіть у випадку відмови деяких вузлів мережі.

3. Шифрування та безпека: IPFS надає можливість шифрувати дані перед їхнім збереженням, що забезпечує конфіденційність і безпеку інформації.

Щодо Pinata, це хмарний сервіс, який надає інструменти для керування та зберігання даних в IPFS мережі. Pinata надає зручний інтерфейс для завантаження, керування та розповсюдження даних через IPFS. Використання Pinata може спростити роботу з IPFS, забезпечуючи надійне зберігання та управління даними для вашого застосунку.

3.1.4 React.js

Використання React в цьому випадку може бути дуже зручним з кількох причин:

1. Компонентна архітектура: React базується на компонентах, що робить його ідеальним для розділення функціональності вашого застосунку на невеликі, повторно використовувані частини. Ви можете створювати окремі компоненти для взаємодії з вашим смарт-контрактом, завантаження файлів на IPFS, відображення даних та багато іншого.

2. Ефективне управління станом: React надає зручні інструменти для управління станом вашого застосунку. Ви можете легко визначити, які дані повинні зберігатися в стані вашого застосунку та як вони повинні бути оновлені при зміні. Це особливо корисно при взаємодії з розумними

контрактами та IPFS, де стан застосунку може змінюватися під час операцій з мережею.

3. Багатофункціональність: За допомогою додаткових бібліотек для React, таких як ethers-react для взаємодії з Ethereum або IPFS API для роботи з IPFS, ви можете легко інтегрувати ці функції в ваш застосунок. React надає зручний спосіб створювати і керувати функціональністю вашого застосунку.

3.2 Backend архітектура системи

3.2.1 Структура зберігання даних в смарт-контракті

В смарт контракті треба писати максимально ефективний код з розрахунку на те що від цього буде залежати скільки буде коштувати взаємодія з ним. Отже дані я зберігаю у наступних структурах :

```
mapping(address => Access[]) public accessList;
mapping(address => Report[]) public reports;
mapping(address => mapping(address => bool)) public ownership;
mapping(address => mapping(address => bool)) public previousAccess;

struct Report{
    string reportUrl;
    string name;
    uint256 date;
    string feedback;
}
```

Рис 3.1 - структура зберігання даних

В полі reports зберігається пари ключ-значення у форматі “адреса-масив структур Report”.

Інші структури даних мають схожу модель, таким чином за адресою гаманця користувача можна дістати посилання на звіти які він завантажував а також перевірити його доступ до них.

Adress	File-1	File-2
0x5fdEf9476FDe30cDa6a84f9207D475748Ddfa7F9	https://white-genuine-moose-392.mypinata.cloud/ipfs/QmVCYRDjsa7CG42Rjuh4Zkgxr2iiMgewehwAbTvANm58tC	https://white-genuine-moose-392.mypinata.cloud/ipfs/QmShd2CeTEH2QVnKx2YejkZUP3FLZLLusdhLeMiCFSjYeG
0xD4d775d3C45a7c0628F5422DC6a87ebcB6aF0bC6	https://white-genuine-moose-392.mypinata.cloud/ipfs/QmShd2CeTEH2QVnKx2YejkZUP3FLZLLusdhLeMiCFSjYeG	

Рис 3.2 - структура зберігання даних (таблиця)

3.2.2 Модель передачі доступу до даних

Щоби поширити або забрати доступ до звітів використовуються наступні методи :

```
function allow(address _user) external {
    ownership[msg.sender][_user]=true;
    if (previousAccess[msg.sender][_user]==true){ // if user once has given access
        for(uint i=0; i<accessList[msg.sender].length; i++){
            if (accessList[msg.sender][i].user==_user){
                accessList[msg.sender][i].hasAccess=true;
            }
        }
    } else {
        accessList[msg.sender].push(Access(_user, true));
        previousAccess[msg.sender][_user]=true;
    }
}

function disallow(address _user) external {
    ownership[msg.sender][_user]=false;
    for(uint i=0; i<accessList[msg.sender].length; i++){
        if (accessList[msg.sender][i].user==_user){
            accessList[msg.sender][i].hasAccess=false;
        }
    }
}
```

Рис 3.3 - структура передачі доступу до даних

Тобто для користувача якому надається доступ до звітів створюється нова пара в структурі яку описували вище, але якщо користувач вирішив забрати доступ у іншого користувача, а потім знову надати його, то для цього користувача не буде створюватися нова пара значень, спочатку буде перевірка чи мав вже користувач колись доступ, і якщо так, то значення hasAcess стане true. Це приклад того як варто економити пам’ять та ресурси блокчейну.

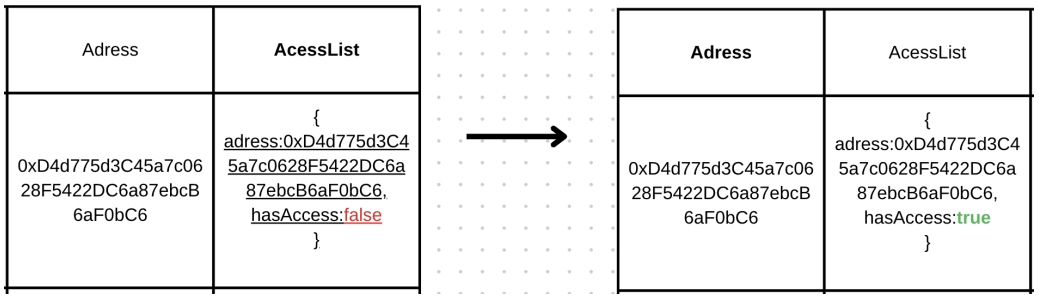


Рис 3.4 - структура передачі доступу до даних(візуалізація)

3.3 Frontend архітектура системи

3.3.1 Підключення до смарт контракту

Підключення до смарт-контракту відбувається в кілька етапів:

1. Створюється функція, яка встановлює з'єднання з веб-провайдером Ethereum за допомогою `***ethers.providers.Web3Provider***`. Якщо провайдер доступний, код встановлює прослуховувачі подій `***chainChanged***` та `***accountsChanged***`, які відслідковують зміни мережі Ethereum та активних облікових записів гаманця. При будь-якій зміні, сторінка перезавантажується і користувач вже бачить нові дані.
2. Функція також відправляє запит на підключення до облікового запису користувача за допомогою `***walletProvider.send('eth_requestAccounts',[])***`. Після успішного підключення вона отримує екземпляр `Signer` для підписування транзакцій.
3. За допомогою підписника, код отримує адресу облікового запису та встановлює змінні `***walletAccount***` та `contract`.
4. Якщо веб-провайдер недоступний, виводиться повідомлення "Please install Metamask".
5. Далі змінні account та contract використовуються в коді для виклику функцій смарт контракту та авторизації.

3.3.2 Опис наявних компонентів та ресурсів

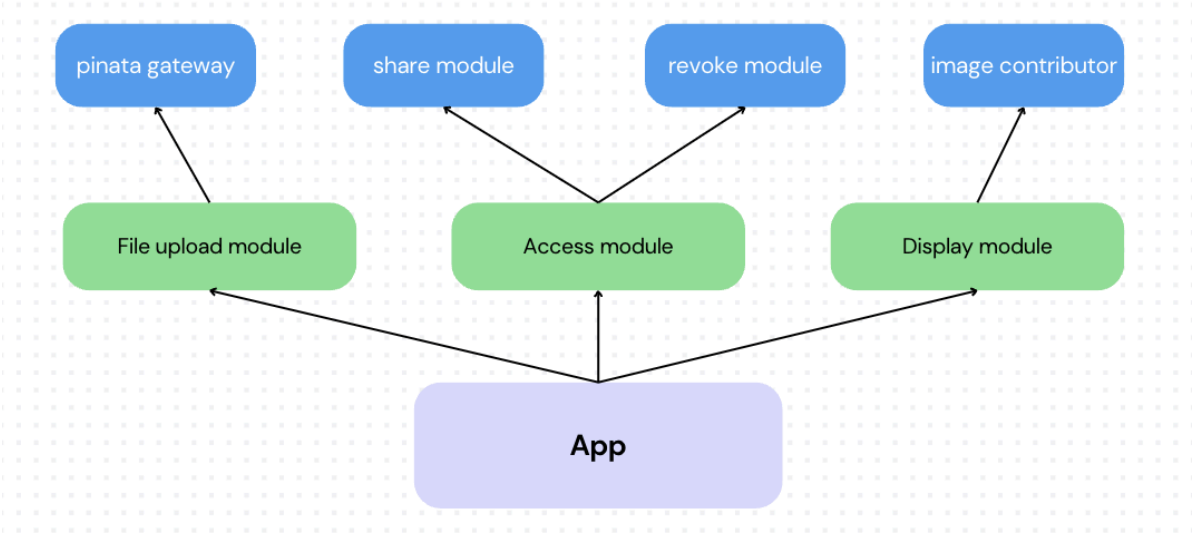


Рис 3.5 - Модулі в React.js

3.3.2.1 Аутентифікація

Аутентифікація та авторизація повністю лягає на гаманець Metamask. Як тільки ви заходите на сторінку, вас автоматично буде перенаправлено на сторінку аутентифікації для цифрового підпису та підтвердження що ви є власником вашого гаманця.

3.3.2.2 IPFS file upload модуль

Основна логіка модуля полягає в тому що ми взаємодіємо з API Pinata, в даному випадку це ендпоінт https://api.pinata.cloud/pinning/pinFileToIPFS, в який параметрами в ми

передаємо ключі для авторизації до проекту та сам файл отриманий від користувача. Після завантаження отриманий hash файлу з IPFS додається в блокчейн через взаємодію зі смарт-контрактом за допомогою виклика функції *contract.add(account,ImgUrl)*.

```
const handleSubmit = async (e) => {
  e.preventDefault();
  if (file) {
    try {
      const formData = new FormData();
      formData.append("file", file);

      const resFile = await axios({
        method: "post",
        url: "https://api.pinata.cloud/pinning/pinFileToIPFS",
        data: formData,
        headers: {
          pinata_api_key: `1997282f8309bc54284d`,
          pinata_secret_api_key: `eb9d25754d55451e697850ddaf74e9e30a6bd3a8e5cd3f909d6bd5695827cd2c`,
          "Content-Type": "multipart/form-data",
        },
      });
      const ImgUrl = `https://gateway.pinata.cloud/ipfs/${resFile.data.IpfsHash}`;
      contract.add(account,ImgUrl);
      setFileName("No image selected");
      setFile(null);
    } catch (e) {
      alert("Unable to upload image to Pinata. Look into logs to see the problem.");
    }
  }
};
```

Для забезпечення надійності доступу до даних ці ключі необхідно зберігати в environment файлах, доступитися до яких можна за допомогою бібліотек таких як dotenv. Перевірити завантажені в IPFS файли та інформацію про них можна за адресою https://app.pinata.cloud/pinmanager, якщо ви менеджер проекту.

3.3.2.3 Модулі передачі доступу до даних

Доступ до даних від користувача до інших поширюється на всі звіти в контексті одного гаманця. В модальному вікні поширення доступу все просто і зрозуміло , користувач вводить адресу отримувача, яка потім валідується і викликається смарт-контракт :

```
const sharing = async () => {
  try{
    const address = document.querySelector(".address").value;
    await contract.allow(address);
  }catch(e){
    setError("Please enter a valid address");
  }
};
```

В модальному вікні відміни доступу спочатку викликається смарт-контракт, беруться всі адреси користувачів із доступом та завантажуються на сторінку, після чого користувач може обрати у якого користувача відібрати доступ до звітів та знову викликати смарт-контракт для позначення що інший користувач тепер hasAccess:false :

```
const revoke = async () => {
  try{
    const selectElement = document.querySelector("#selectAddress");
    const address = selectElement.value;
    console.log(address);
    await contract.disallow(address);
  }catch(e){
  }
}

useEffect(() => {
  const accessList = async () => {
    const addressList = await contract.shareAcess();
    let select = document.querySelector("#selectAddress");
    for (let i = 0; i < addressList.length; i++) {
      let accessMember = addressList[i];
      if(accessMember[1] == true){ //check if has access
        let option = document.createElement("option");
        option.textContent = accessMember[0]; //getting address here
        option.value = accessMember[0];
        select.appendChild(option);
      }
    }
  };
  contract && accessList();
}, [contract]);
```

3.3.2.4 Модуль презентації даних

В цьому модулі користувач може отримати масив звітів за адресою яку уведе, або свої звіти якщо залишить поле пустим. В цьому компоненті використовується вбудований компонент з бібліотекою парсингу pdf файлів, адже зазвичай звіти мають далеко не одну сторінку.

```
const getdata = async () => {
  let dataArray;
  const Otheraddress = document.querySelector("#address").value;
  console.log(Otheraddress);
  try {
    if (Otheraddress) {
      dataArray = await contract.display(Otheraddress);
    } else {
      dataArray = await contract.display(account);
    }
  } catch (e) {
    alert("You don't have access.");
    return;
  }
  const isEmpty = Object.keys(dataArray).length === 0;
  if (!isEmpty) {
    const str = dataArray.toString();
    const str_array = str.split(",");
    const images = str_array.map((item, i) => {
      return (
      <a className="overflow-hidden" href={item} key={i} target="_blank">
        <ImageViewer pdfURL={item}/>
      </a>
      );
    });
    setData(images);
  } else {
    alert("No image to display");
  }
};
```

3.4 Тестування програми і результати її виконання

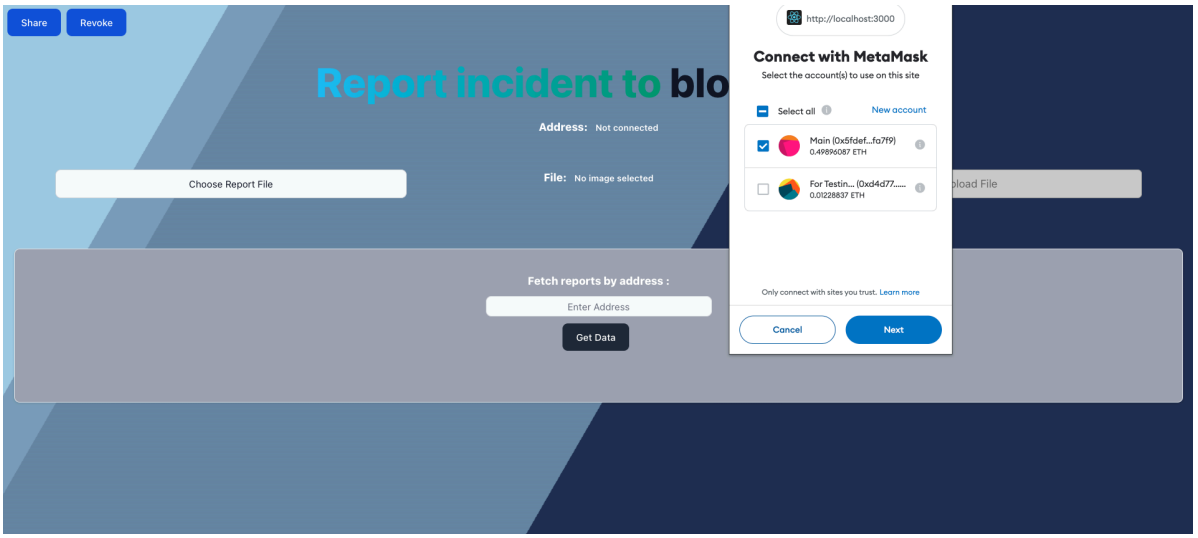
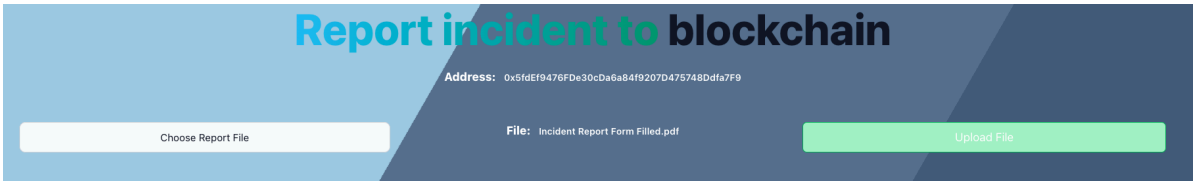
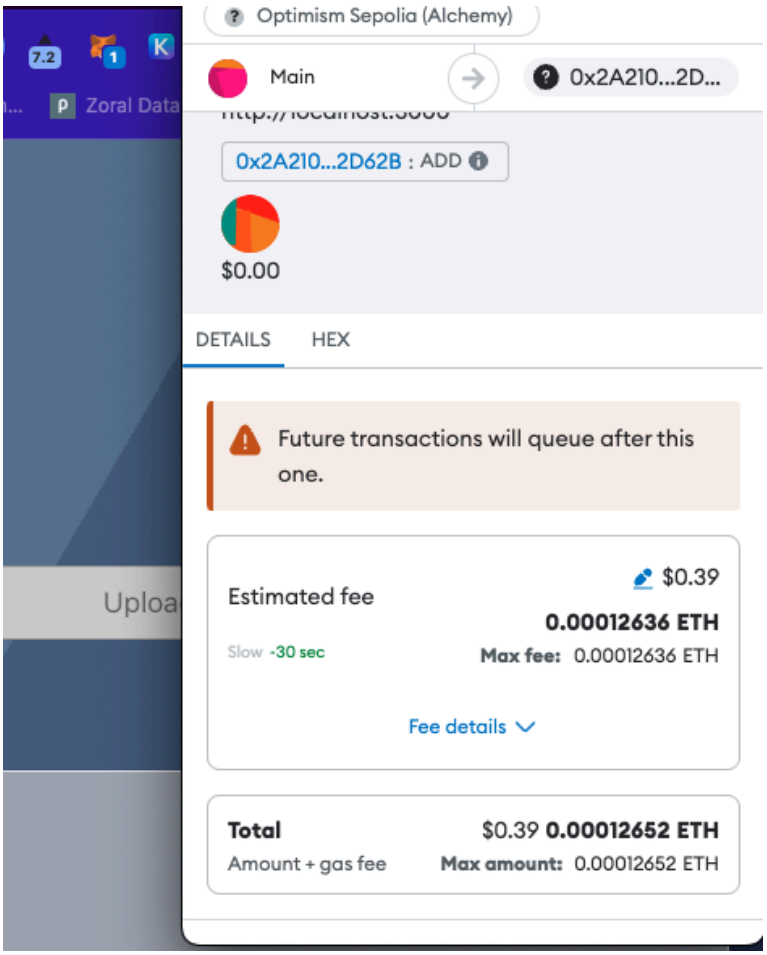


Рис 3.6 - авторизація користувача

Після успішної авторизації користувач може завантажити звіт.

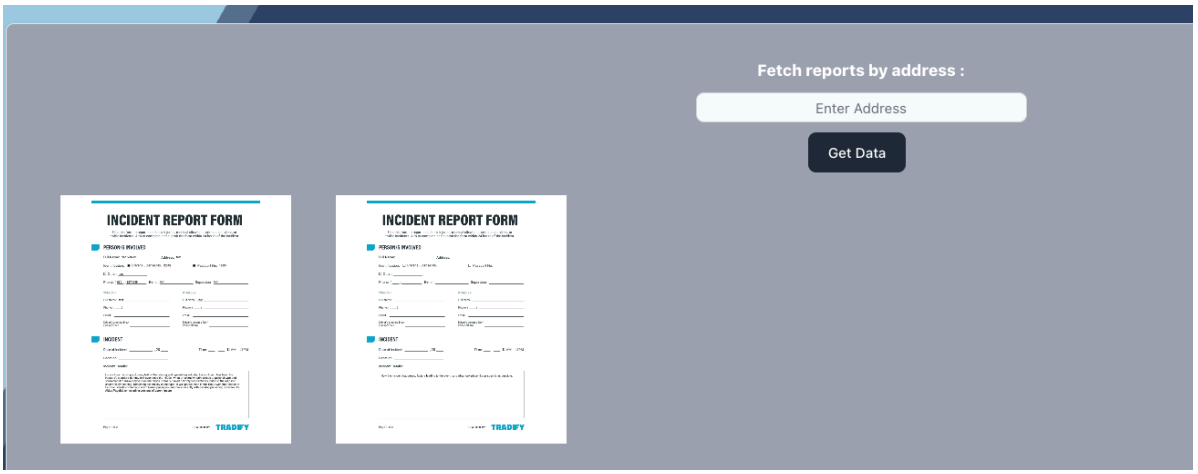


При підтвердженні відкриється вікно для сплати комісії за транзакцію:

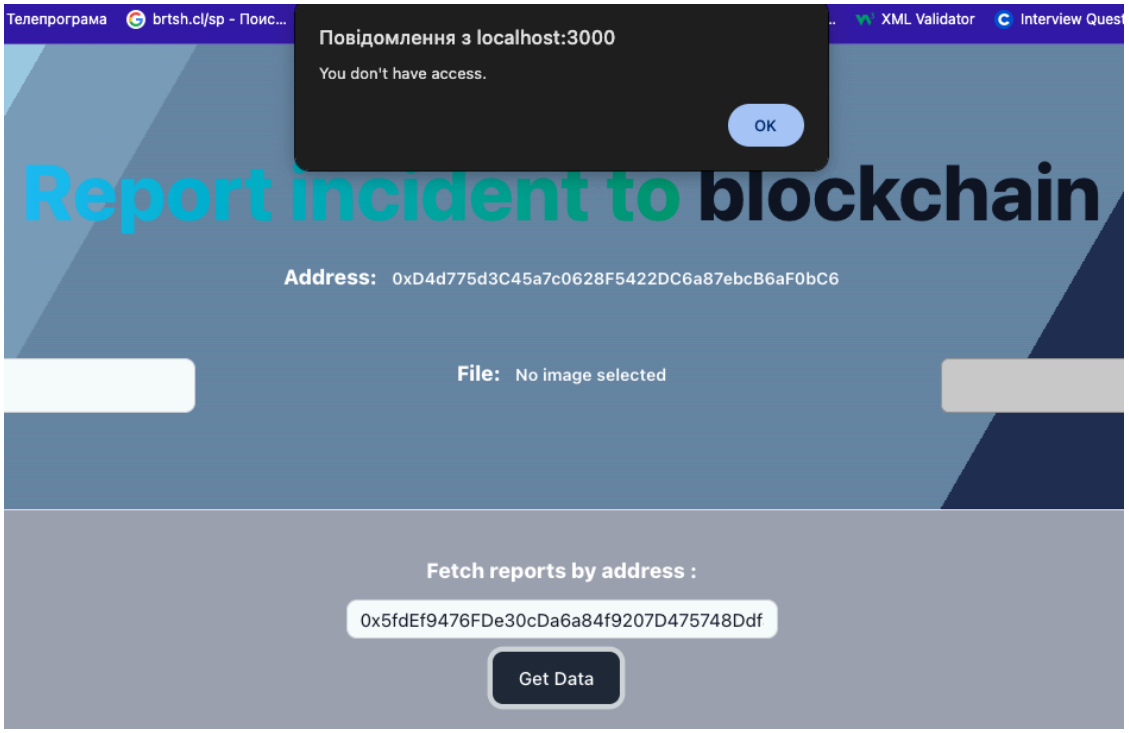


Комісія досить висока, через те що це тестова мережа, з тестовими грошима, в реальній мережі така транзакція обійшлась в пару центів зважаючи на останні оновлення в мережі Ethereum.

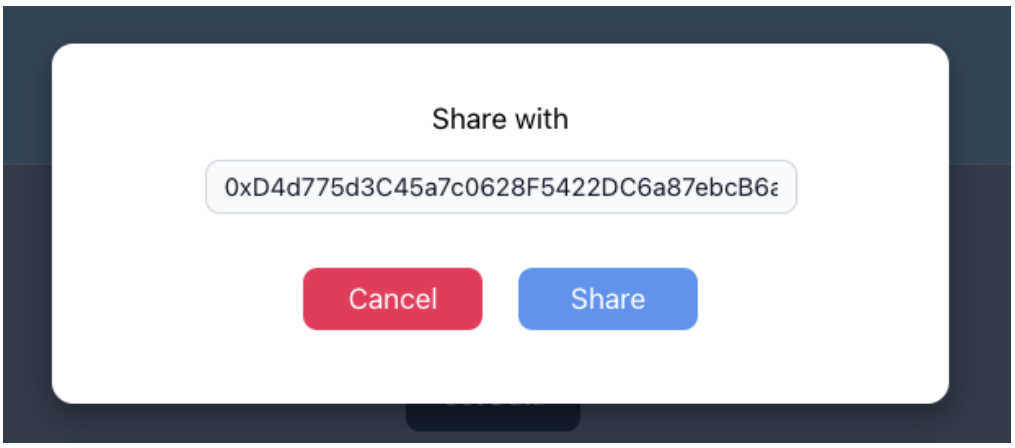
Тепер користувач може переглянути звіт який він завантажив.



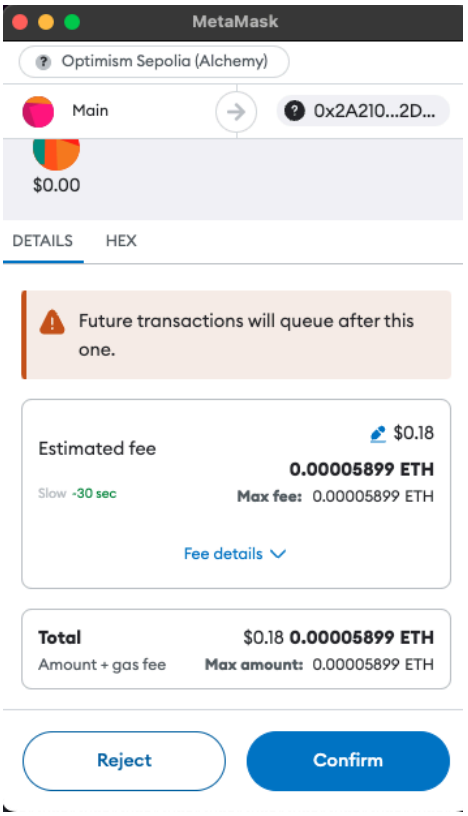
Чудово, це у нас користувач номер 1 за адресою **0x5fdEf9476FDe30cDa6a84f9207D475748Ddfa7F9** , тепер нехай інший користувач номер 2, з адресою **0xD4d775d3C45a7c0628F5422DC6a87ebcB6aF0bC6** захоче отримати його звіти. Він авторизується, вводить адресу 1 користувача, натискає на Get Data та отримує повідомлення :



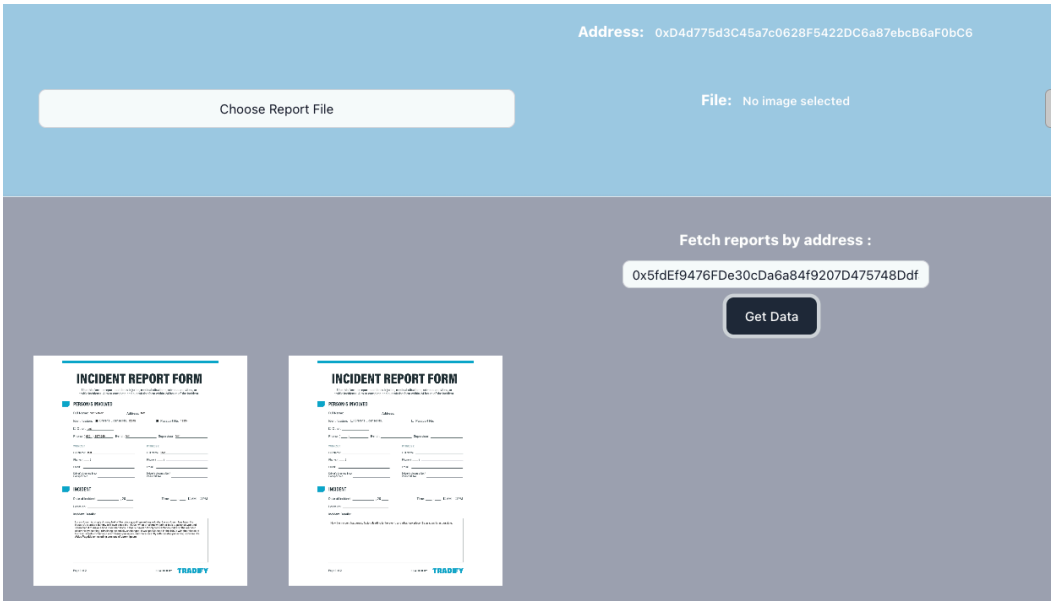
Нехай користувач 1 надає доступ користувачу 2 :



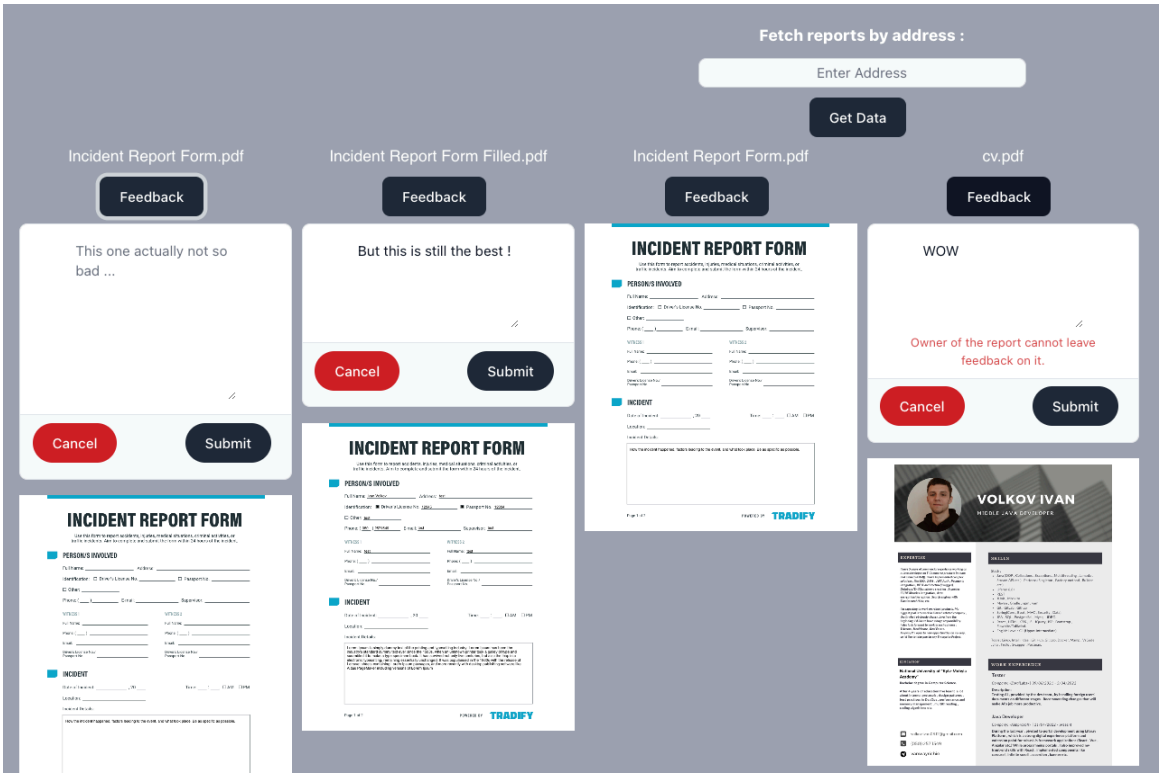
Сплачує транзакцію.



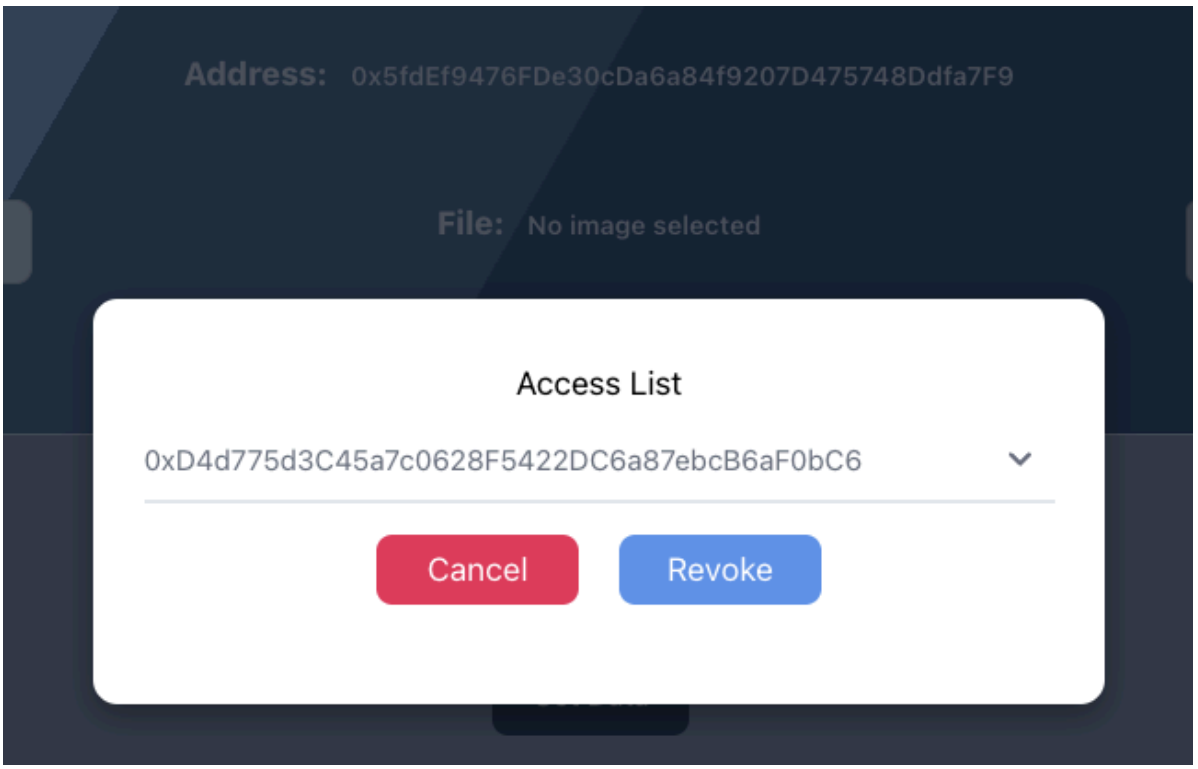
Тепер користувач 2 може подивитися звіти які йому “надіслав” користувач 1.



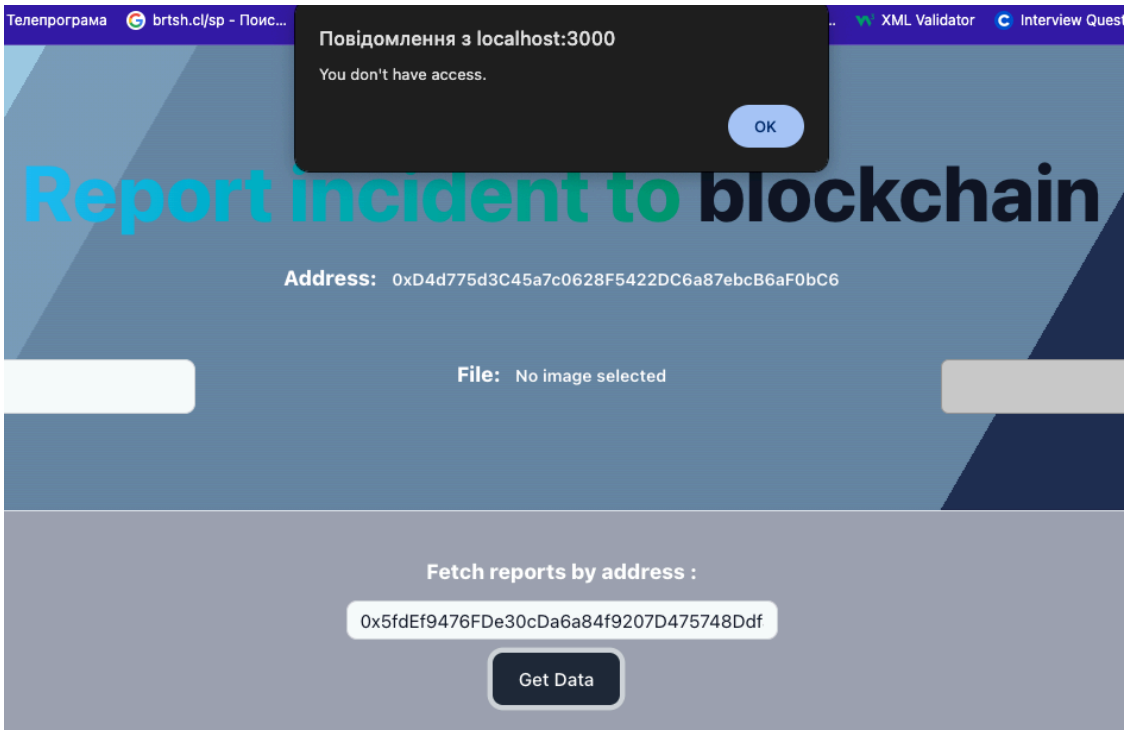
А також залишати відгуки до звітів (причому може тільки той кому надав доступ попередній користувач) :



Користувач 1 забирає доступ та сплачує транзакцію.



Тепер користувач 2 знову не має доступу до звітів



Висновки

Отже поставлена задача була виконана шляхом використання інструментів децентралізації. Звіти тепер можна зберігати та передавати через блокчейн і більше не переживати про те що звіт/колекцію звітів буде втрачено назавжди адже копія звітів зберігається на багатьох комп’ютерах одночасно. Одночасно з тим система зручна адже фронт енд архітектура на React дає змогу розбивати функціонал на модулі, в неї було інтегровано багато зручних функцій, як наприклад пошук по датах або ключових словах, обговорення та система зворотнього зв’язку між регулятором та компанією або користувачем, а також структура зберігання даних(пари ключ значення) дає змогу відстежувати хто надавав звіт та інтегрувати автоматичне або ручне(пересилання криптовалюти) нагородження користувача за звітування за допомогою смарт-контракту та мови програмування Solidity.

Список використаної літератури

1. The European Union Agency for Cybersecurity (ENISA): https://www.enisa.europa.eu/topics/incident-reporting https://www.enisa.europa.eu/news/enisa-news/enisa-report-on-blockchain-technology-and-security
2. Blockchain-based Incident Reporting System for Patient Safety and Quality in Healthcare: https://www.biihealthtech.com/wp-content/uploads/2021/08/5.Blockchain-based-Incident-Reporting-System-for-Patient-Safety-and-Quality-in-Healthcare.pdf
3. CoinDesk, an award-winning media outlet that covers the cryptocurrency industry, about the need in blockchain based report systems: https://www.coindesk.com/policy/2023/07/28/us-listed-crypto-firms-will-need-to-report-cybersecurity-breaches/
4. Pinata IPFS Documentation: https://docs.pinata.cloud/introduction
5. Alchemy docs on deploying smart contracts to different blockchains https://docs.alchemy.com/
6. What To Include in a Cyber Incident Report https://www.upguard.com/blog/cyber-incident-reporting#:~:text=Cyber%20incident%20reporting%20is%20when,affected%20customers%2C%20business%20partners%2C%20and
7. http://www.socialalphafoundation.org/wp-content/uploads/2022/01/saf-blockchain-report-final-2022.pdf
8. Best practises in using Solidity with IPFS : https://itnext.io/storage-dapp-using-solidity-and-ipfs-62cf371b77dc
9. React and ethers.js to communicate with smart contracts: https://docs.ethers.org/v4/cookbook-react.html
10. Flowbite/Tailwind front end frameworks documentation:

https://flowbite.com/docs/getting-started/quickstart/

Додаток А

Додаток Б