Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних Наук
(повна назва)

Кафедра Програмної Інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів стиснення зображень
у блокчейн сховищах
(тема)

Виконав:
студент (ка) 2 курсу, групи ІПЗм-22-3

Олійник А.Є.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного
Забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-наукова

Керівник доц. Кириченко І.В.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних Наук
Кафедра Програмної Інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
Спеціалізація Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Олійнику Артему Євгенійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження методів стиснення зображень в блокчейн сховищах»

Затверджена наказом університету від «29» березня 2024 р. № 250 Ст.

2. Термін подання студентом роботи до екзаменаційної комісії 13.06.2024

3. Вихідні дані до роботи *Опис досліджуваних методів стиснення зображень, вимоги до розробки схеми зберігання та доступу до зображень для проведення досліджень за обраною предметною областю, мови програмування C#, Solidity, технології .NET 8.0, RazorPages, ImageMagick, Nethereum, системи управління базами даних IPFS, середовища розробки Rider, Ganache, Truffle*

4. Перелік питань, що потрібно опрацювати в роботі
аналіз та порівняння існуючих методів стиснення зображень, вибір підходящих методів стиснення для дослідження, аналіз зберігання та доступу до зображень у IPFS, написання програмних рішень для стискання зображень та зберігання хешів у блокчейні, проведення експериментів та аналіз отриманих результатів.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	29.03.2024	*виконано*
2	Виявлення проблематики галузі	31.03.2024	*виконано*
3	Постановка задачі	03.04.2024	*виконано*
4	Програмна реалізація	03.04.2024 – 10.05.2024	*виконано*
6	Проведення порівняльного експерименту	11.05.2024	*виконано*
9	Підготовка пояснювальної записки	03.04.2024 – 15.05.2024	*виконано*
10	Підготовка презентації та доповіді	15.05.2024 – 20.05.2024	*виконано*
11	Нормоконтроль	22.05.2024	*виконано*
12	Рецензування	25.05.2024	*виконано*
13	Занесення диплома в електронний архів	27.05.2024	*виконано*
14	Попередній захист	27.05.2024	*виконано*
15	Допуск до захисту у зав. кафедри	12.06.2024	*виконано*
	Захист магістерського дослідження	13.06.2024	*виконано*

Дата видачі завдання 20 січня 2024р.

Студент (ка) _____ Олійник А.Є.
(підпис)

Керівник роботи _____ доц. Кириченко І.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи магістра містить: 78 с., 18 рис., 4 табл., 21 джерел.

БЛОКЧЕЙН, СТИСНЕННЯ ЗОБРАЖЕНЬ, СХОВИЩА ДАНИХ, IPFS, GANACHE, ГРАФІКА.

Об’єктом дослідження є методи стиснення зображень та їх інтеграція з блокчейн сховищами.

Метою роботи є проведення дослідження продуктивності методів стиснення зображень для зберігання у блокчейн сховищах на платформі .NET.

Методами розробки та проектування є аналіз проблемної області дослідження, вибір методів стиснення зображень для дослідження шляхом аналізу їх продуктивності, якості стиснення та інших критеріїв за допомогою багатокритеріального аналізу та прийняття рішень.

У результаті кваліфікаційної роботи було проведено дослідження предметної галузі, розроблено програму для проведення дослідження методів стиснення зображень та їх інтеграції з блокчейн сховищами на платформі .NET.

BLOCKCHAIN, IMAGE COMPRESSION, DATA STORAGE, IPFS, GANACHE, GRAPHICS.

The object of the research is methods of image compression and their integration with blockchain storage.

The purpose of the work is to conduct research on the performance of image compression methods for storage in blockchain repositories on the .NET platform.

The methods of development and design include the analysis of the research problem area, the selection of image compression methods for research by analyzing their performance, compression quality, and other criteria using multi-criteria analysis and decision-making.

As a result of the qualification work, research in the subject area was conducted, and a program for investigating image compression methods and their integration with blockchain repositories on the .NET platform was developed.

Заява щодо самостійного виконання кваліфікаційної роботи та можливості її публікації в електронному архіві відкритого доступу ElArKhNURE.

Я, Олійник Артем Євгенійович, студент(ка) гр. ІПЗм-22-3, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів стиснення зображень в блокчейн сховищах», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений(на) з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

JPEG – Joint Photographic Experts Group

PNG – Portable Network Graphics

BMP – Bitmap

TIFF – Tagged Image File Format

GIF – Graphics Interchange Format

LZ77 – Lempel-Ziv 1977

HTTP – Hypertext Transfer Protocol

PSNR – Peak Signal-to-Noise Ratio

IPFS – Inter-Planetary File System

LZW – Lempel–Ziv–Welch

DCT – Discrete cosine transform

RGB – Red-Green-Blue

ВСТУП

Сучасний науковий прогрес активно просуває технології, випереджаючи їхній розвиток і впровадження нововведень. Раніше для зберігання великого обсягу зображень доводилося витрачати значний час і ресурси. Тепер майже кожна компанія спрямована на поліпшення процесу зберігання шляхом інтеграції систем стиснення зображень у свої продукти.

Системи стиснення здатні працювати з різними типами даних, такими як зображення, звук, відео, текст тощо. Вони відрізняються одна від одної своєю архітектурою, типом пам’яті, тощо. Актуальність даної теми полягає в тому, що за останні роки в галузі зберігання даних зросла потреба у надійних і безпечних методах забезпечення конфіденційності та цілісності інформації. В контексті розвитку технологій зображень, виникає велика потреба у збереженні та обміні цими даними в безпечних та надійних умовах. Таким чином, вивчення методів стиснення зображень для їх зберігання у блокчейн-сховищі має велике практичне значення у сучасному інформаційному середовищі. У цьому дослідженні вирішено дослідити методи стиснення зображень для їх ефективного зберігання у блокчейн-сховищі оскільки технологія блокчейну [1] пропонує ряд переваг для зберігання даних, таких як підвищена безпека, надійність та масштабованість.

На сьогоднішній день існує багато доступних систем стиснення зображень, які відрізняються за якістю в залежності від алгоритму. Якість стиснення прямо пропорційна якості алгоритму, який, у свою чергу, залежить від розміру, типу та обсягу даних, а також від типу системи, в якій він застосовується.

Метою роботи є визначення найбільш ефективного методу стиснення зображень для зберігання в блокчейн сховищах з точки зору розміру файлу, якості зображення та продуктивності передачі даних.

Для досягнення цієї мети необхідно вирішити такі завдання:

- здійснити аналіз та порівняння існуючих методів стиснення зображень;
- визначити можливий вплив обраного методу стиснення на якість та розмір зображень;

- розробити програмне забезпечення для проведення експерименту зі стисненням зображень;
- виміряти та аналізувати розмір файлів та якість стиснених зображень;
- провести експерименти щодо часу передачі стиснених зображень у IPFS;
- надати рекомендації щодо вибору оптимального методу стиснення зображень для конкретних сценаріїв використання.

Об'єктом дослідження є ефективність методів стиснення зображень, які використовуються для зберігання у блокчейн сховищах. Предметом дослідження є процес стиснення зображень та його вплив на розмір файлів та якість зображення.

Методами дослідження є проведення експериментів зі стисненням зображень різними методами та аналіз результатів за допомогою спеціалізованих програмних інструментів.

Отримані результати можуть бути використані для покращення процесу зберігання зображень у блокчейн сховищах, а також можуть послужити основою для подальших досліджень у галузі стиснення даних у децентралізованих системах.

Результати даної кваліфікаційної роботи було опубліковано в журналі «Біоніка інтелекту», який включено до Переліку наукових фахових видань України, категорія «Б», технічні науки (затверджено наказом МОНУ від 02.07.2020 р. № 886).

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз проблемної області дослідження

З розвитком цифрових технологій обсяг інформації, яку необхідно зберігати та передавати, постійно зростає. Однією з найбільш поширених форм інформації є зображення, що використовуються в різних галузях, включаючи медицину, науку, мистецтво та комерцію. Однак, зберігання та передача зображень часто є складним завданням через їх великий розмір та вимоги до пропускної здатності мережі. У цьому контексті дослідження методів стиснення зображень стає надзвичайно актуальним.

В той же час, технологія блокчейн, яка забезпечує децентралізоване, безпечне та прозоре зберігання інформації [2], стає все більш популярною для зберігання різноманітних даних.

Блокчейн-сховища – це децентралізовані системи зберігання даних, які використовують технологію блокчейну для забезпечення безпеки і надійного зберігання даних [3]. Стиснення зображень може бути корисним для блокчейн-сховищ, оскільки воно може допомогти зменшити розмір даних, що зберігаються, і покращити ефективність обробки зображень.

При виборі методу стиснення для блокчейн-сховищ слід враховувати такі фактори:

- блокчейн-сховища використовують децентралізовані мережі, які мають обмежену пропускну здатність [4]. Це означає, що важливо використовувати методи стиснення, які забезпечують невеликий розмір стиснених даних.
- вибір методу стиснення має забезпечувати баланс між зменшенням розміру файлу і збереженням достатньої якості зображення, особливо коли йдеться про стиснення з втратами. Метрики, такі як PSNR, допомагають оцінити якість стиснених зображень і знайти оптимальний компроміс між розміром та якістю [5].

Блокчейн-сховища, такі як IPFS, надають можливість зберігати файли з високим рівнем безпеки та стійкості [2]. IPFS дозволяє створювати унікальний відбиток (хеш) для кожного файлу, що додається до мережі, забезпечуючи надійну ідентифікацію та доступ до збережених даних [6].

Однак, зберігання великих зображень у блокчейн-сховищах може бути неефективним через обмеження на пропускну здатність та вартість зберігання [7]. Тому дослідження методів стиснення зображень перед їх зберіганням у блокчейн-сховищах, є важливим і актуальним завданням.

У даному дослідженні розглядаються методи стиснення зображень для зберігання у блокчейн-сховищах на прикладі IPFS. Основна увага приділяється порівнянню різних алгоритмів стиснення зображень за такими критеріями, як розмір файлу, якість зображення за метрикою PSNR, час завантаження в IPFS та пропускна здатність мережі. Таке порівняння допоможе визначити оптимальні методи стиснення зображень для їх ефективного зберігання у блокчейн-сховищах, забезпечуючи при цьому високу якість зображень та ефективне використання мережевих ресурсів.

Методи стиснення зображень можна розділити на два основні типи:

– стиснення з втратами – метод, який дозволяє значно зменшити розмір файлу за рахунок втрати якості зображення. Цей метод дозволяє досягти значного зменшення розміру файлу без помітних втрат якості;
– стиснення без втрат – метод, який дозволяє зберегти початкову якість зображення. Однак, цей метод, як правило, не дозволяє досягти такого значного зменшення розміру файлу, як стиснення з втратами.

Плюси втратного стискання:

– висока ефективність. Втратне стиснення може забезпечити значно більший ступінь стиснення, ніж безвтратне. Це може призвести до значної економії місця на диску, підвищення пропускної здатності мережі та покращення ефективності обробки зображень.

Мінуси втратного стискання:

– втрата якості. Втратне стискання передбачає видалення деякої інформації з зображення. Це може призвести до втрати деталей, зміни кольору або контрасту;
– неможливість відновлення даних. Втративши інформацію, яку було видалено при стисканні, її неможливо відновити.

Втратне стиснення широко застосовується для обробки зображень, де незначна втрата деталізації є допустимою і не впливає суттєво на їх сприйняття. Використання методів втратного стиснення, дозволяє значно зменшити розмір файлу за рахунок часткового зменшення якості зображення, що є прийнятним компромісом у багатьох випадках. Ці методи базуються на алгоритмах, що відкидають або узагальнюють деякі деталі зображення, які менш помітні для людського ока, зберігаючи при цьому загальну візуальну якість.

Плюси безвтратного стискання:

– збереження якості. Безвтратне стискання не призводить до втрати інформації з зображення. Це означає, що стиснене зображення буде повністю відповідати оригінальному.

Мінуси безвтратного стискання:

– низька ефективність. Безвтратне стиснення зазвичай забезпечує нижчий ступінь стиснення, ніж втратне. Це означає, що для зберігання стиснених зображень може знадобитися більше місця на диску або пропускної здатності мережі.

Безвтратне стискання використовується в різних областях, де збереження якості зображень є вирішальним аспектом. Наприклад, у медичній сфері або в обробці зображень для друку, важливо зберегти всі важливі деталі та точність, тому безвтратні методи стискання є невід'ємною частиною процесу стиснення зображень. Використання безвтратного стискання дозволяє зменшити обсяг даних, зберігаючи при цьому високий рівень якості, що робить його незамінним інструментом у сучасній обробці зображень. (див. табл. 1.1).

Таблиця 1.1 – Порівняння втратного і безвтратного стискання (таблиця виконана самостійно)

Характеристика	Втратне стискання	Безвтратне стискання
Ступінь стиснення	Вищий	Нижчий
Якість зображення	Може бути втрачена	Зберігається
Відновлення даних	Неможливе	Можливе
Застосування	Зображення, для яких втрата якості не є критичною	Зображення, для яких втрата якості є не допустимою

Вибір методу стиснення залежить від конкретного застосування. Для зображень з високою деталізацією, для яких втрата якості не є допустимою, слід використовувати безвтратне стиснення. Для зображень, для яких важлива пропускна здатність або розмір файлу, можна використовувати втратне стиснення

1.2 Аналіз алгоритмів стиснення зображень

Після того, як ми зрозуміли актуальність дослідження методів стиснення зображень у контексті їх зберігання в блокчейн-сховищах, наступним кроком є аналіз доступних алгоритмів стиснення. Розглянемо основні методи, які використовуються для зменшення розміру зображень без втрати якості та з втратами, щоб визначити їх ефективність у контексті наших досліджень. Цей аналіз допоможе нам обрати оптимальні методи стиснення для нашого проекту, забезпечуючи максимальну ефективність при збереженні якості зображень.

У дослідженні розглядаються наступні алгоритми:

а) стиснення без втрат:

1) PNG;

2) TIFF;

3) GIF.

б) стиснення з втратами:

1) JPEG;

2) WEBP.

1.2.1 Метод стиснення PNG

Метод стиснення PNG є форматом зображень, який використовується для зберігання растрових зображень [8]. Однією з ключових особливостей PNG є можливість стиснення без втрат якості. Для цього використовується алгоритм стиснення Deflate.

Deflate – це алгоритм стиснення, що використовується в форматі PNG для зменшення розміру файлу без втрати якості. Він базується на комбінації алгоритму LZ77 та кодування Хаффмана [9].

Алгоритм LZ77 використовується для заміни повторюваних послідовностей байтів на зворотні посилання на знайдені раніше послідовності в тексті. Цей процес ґрунтується на вікні ковзання, яке зберігає копії останніх даних. Замість зберігання повної копії, використовується посилання на позицію та довжину послідовності.

Після застосування LZ77 до даних, отримані символьні послідовності кодуються за допомогою кодування Хаффмана, яке стискає часто повторювані символи у менші біти та рідко повторюванні символи у більші.

Початковий етап полягає у визначенні ймовірностей появи кожного символу у зображенні. У контексті стиснення зображень ці символи можуть бути представлені, наприклад, значеннями пікселів або іншими кодами, що представляють кольори.

Для обчислення ймовірностей використовується формула 1.1:

$$p_i = \frac{n_i}{N}, \tag{1.1}$$

де p_i – ймовірність появи символу i,

n_i – кількість входжень символу i у зображенні,

N – загальна кількість символів у зображенні.

Після обчислення ймовірностей будується дерево Хаффмана. Для кожного символу будується листок дерева з вагою, що відповідає ймовірності появи символу. Внутрішні вузли мають вагу, яка є сумою ваг дітей. Після побудови дерева вузлам надаються ваги, відповідні їх ймовірностям. Для обчислення ваги листка використовується формула 1.2:

$$\omega_i = p_i \cdot N, \tag{1.2}$$

де ω_i – вага листка для символу i,

p_i – ймовірність появи символу i,

N – загальна кількість символів у зображенні.

Для призначення кодів Хаффмана кожному символу призначається унікальний код, що визначається шляхом від кореня дерева до відповідного листка. Код складається з послідовності бітів, де 0 вказує на ліве відгалуження, а 1 – на праве.

Після цього зображення кодується, замінюючи кожен символ його відповідним кодом. Це призводить до створення бітової послідовності, яка представляє стиснене зображення.

Наприклад для побудови дерева Хаффмана для даних «AAAABBBBBCCCCDDD» після аналізу частоти, буде отримано наступні значення:

– A: частота 4 рази, ймовірність появи 4/16;
– B: частота 5 разів, ймовірність появи 5/16;
– C: частота 4 рази, ймовірність появи 4/16;
– D: частота 3 рази, ймовірність появи 3/16.

Після цього визначається вага кожного листка і будується дерево Хаффмана (див. рис. 1.1). Символ «A» кодується як «01», символ «B» кодується як «11», символ «C» кодується як «10» і відповідно символ «D» кодується як «00». Отже,

використовуючи коди Хаффмана можна закодувати текст «AAAABBBBBCCCCDDD» і отримати результат «000011111000001111010 1».

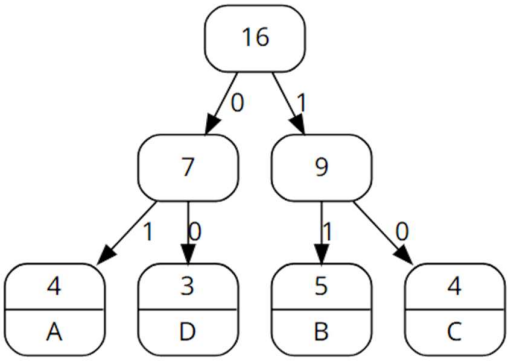

Рисунок 1.1 – Приклад діаграми Хаффмана (рисунок створено самостійно)

Таким чином, метод стиснення PNG забезпечує зменшення розміру файлу без втрати якості. Це дозволяє ефективно зберігати та передавати зображення, зберігаючи всі деталі та точність. Використання таких методів стиснення є критичним для сфер, де важлива висока якість зображення, таких як медична діагностика, наукові дослідження та поліграфія.

1.2.2 Метод стиснення TIFF

Метод стиснення TIFF є форматом зображень, який використовується для зберігання растрових зображень. Однією з ключових особливостей TIFF є можливість стиснення без втрат якості. Для цього використовується алгоритм стиснення LZW.

Алгоритм LZW – це алгоритм стиснення без втрат, що використовується в форматі TIFF для зменшення розміру файлу без втрати якості. Він базується на використанні словника фіксованого розміру, який динамічно оновлюється під час процесу стиснення [10].

Першим етапом є створення словника, який містить всі можливі символи вхідних даних. Під час процесу стиснення, алгоритм шукає найдовшу послідовність символів, яка вже є в словнику, і замінює її на відповідний код словника. Ця послідовність символів потім розширюється на один символ, і нова послідовність додається до словника.

Наприклад, розглянемо стиснення рядка «ABABABA» за допомогою LZW. Спочатку словник містить лише окремі символи «A» з кодом «1» та «B» з кодом «2».

Під час стиснення, алгоритм спочатку знаходить «A» в словнику і виводить відповідний код. Потім він розширює послідовність до «AB», якої ще немає в словнику, тому «AB» додається до словника з кодом «3», а код для «A» виводиться.

Процес продовжується, поки весь рядок не буде стиснутий, і в результаті отримується послідовність кодів словника. Таким чином, вихідний код для рядка «ABABABA» за допомогою алгоритму LZW буде 1, 2, 3, 1, 3 (див. табл. 1.2).

Таблиця 1.2 – Покроковий принцип роботи алгоритму LZW (таблиця виконана самостійно)

Крок	Словник (символ : код)	Вивід алгоритму
1	A : 1, B : 2, AB : 3	1
2	A : 1, B : 2, AB : 3, BA : 4	1, 2
3	A : 1, B : 2, AB : 3, BA : 4, ABA : 5	1, 2, 3
4	A : 1, B : 2, AB : 3, BA : 4, ABA : 5	1, 2, 3, 1
5	A : 1, B : 2, AB : 3, BA : 4, ABA : 5	1, 2, 3, 1, 3

Цей алгоритм ефективний для стиснення даних, які містять повторювані послідовності символів, і дозволяє стискати зображення без втрат. Важливо зазначити, що, хоча LZW є алгоритмом стиснення без втрат, він може бути неефективним для стиснення деяких типів даних. Наприклад, він може не зменшувати розмір файлу, якщо дані містять високий рівень шуму або якщо повторювані послідовності символів є дуже короткими або рідкісними.

1.2.3 Метод стиснення GIF

Метод стиснення GIF є форматом зображень, який використовується для зберігання растрових зображень. Однією з ключових особливостей GIF є можливість стиснення без втрат якості. Для цього використовується алгоритм

стиснення LZW, аналогічний до того, що використовується в TIFF. Однак, варто зазначити, що GIF відрізняється від TIFF декількома нюансами.

По-перше, GIF підтримує анімацію, що дозволяє зберігати кілька зображень в одному файлі.

По-друге, GIF використовує палітру кольорів, що обмежує кількість кольорів, які можуть бути використані в зображенні, до 256 [11]. Це робить GIF менш підходящим для зображень з великою кількістю кольорів, але дуже ефективним для зображень з обмеженою палітрою кольорів, таких як текст, логотипи або прості ілюстрації.

Алгоритм LZW в GIF працює так само, як і в TIFF. Він створює словник, який динамічно оновлюється під час процесу стиснення, замінюючи найдовші послідовності символів, які вже є в словнику, на відповідні коди словника. Це дозволяє GIF стискати зображення без втрати якості. Але також варто зазначити, що цей метод може бути неефективним якщо повторювані послідовності символів є дуже короткими або дані містять високий рівень шуму.

1.2.4 Метод стиснення JPEG

Метод стиснення JPEG є форматом зображень, який використовується для зберігання растрових зображень. Однією з ключових особливостей JPEG є можливість стиснення з втратами якості. Для цього використовується алгоритм стиснення, що базується на дискретному косинусному перетворенні (DCT) [12]. DCT є методом стиснення, що використовується в форматі JPEG для зменшення розміру файлу за рахунок втрати якості. Він базується на перетворенні просторових даних зображення в частотні дані, що дозволяє відкинути частотні компоненти, які менш важливі для сприйняття людським оком. Процес стиснення JPEG можна розбити на 5 основних кроків.

Першим кроком зображення розбивається на блоки розміром 8x8 пікселів, цей процес можна описати формулою 1.3:

$$N = \frac{W \cdot H}{64}, \tag{1.3}$$

де N – загальна кількість блоків,

W – ширина зображення,

H – висота зображення.

Далі кольоровий простір зображення перетворюється з RGB в YCbCr, що відокремлює яскравість (Y) від хроматичності (Cb та Cr) [13].

Третім кроком кожен блок 8x8 пікселів перетворює просторові дані (координати x, y) в частотні дані (компоненти низьких і високих частот) за допомогою DCT. Матриця результатів DCT розраховується за формулою 1.4:

$$F = T \cdot B \cdot T^T, \tag{1.4}$$

де F – матриця результатів DCT,

B – матриця 8x8 пікселів зображення,

T – матриця косинусних перетворень розміром 8x8,

T^T – транспонована матриця перетворень.

Матриця косинусних перетворень розраховується за формулою 1.5:

$$T_{[i,j]} = \alpha(i) \cdot cos\left(\frac{(2j+1) \cdot i \cdot \pi}{16}\right), \tag{1.5}$$

де i – рядок матриці,

j – стовпець матриці,

$\alpha(i)$ – визначається за формулою 1.6:

$$\alpha(i) = \begin{cases} \frac{1}{\sqrt{8}}, & \text{якщо } i = 0 \\ \frac{1}{2}, & \text{якщо } i > 0 \end{cases}. \quad (1.6)$$

Четвертим кроком частотні координати кожного блоку нормалізуються за допомогою таблиці квантування, що зменшує точність високочастотних компонентів. На цьому етапі відбувається втрата інформації. Коефіцієнти ДКП квантуються, тобто округляються до заданих значень. Матриця квантування розраховується за формулою 1.7:

$$B' = \lfloor \frac{F}{Q} \rfloor, \quad (1.7)$$

де B' – квантизовані коефіцієнти DCT,

F – матриця результатів DCT,

Q – матриця квантування,

$\lfloor \cdot \rfloor$ – операція взяття підлоги (округлення вниз),

Матриця Q може бути визначена за допомогою таблиць квантування, які містять значення квантування для різних частотних компонентів та рівнів стиснення [14].

Останнім кроком нормалізовані блоки кодуються за допомогою кодування Хаффмана для подальшого зменшення розміру файлу і збираються разом із заголовком JPEG.

Важливо зазначити, що, хоча JPEG є алгоритмом стиснення з втратами, він дозволяє вибрати баланс між розміром файлу та якістю зображення за допомогою параметра якості. Високі значення якості приводять до меншого стиснення та вищої якості зображення, тоді як низькі значення якості приводять до більшого стиснення та нижчої якості зображення.

1.2.5 Метод стиснення WEBP

WEBP є форматом зображень, розробленим Google, який використовується для зберігання растрових зображень. Однією з ключових особливостей цього формату є можливість стиснення з втратами якості. Для цього використовується алгоритм стиснення VP8.

VP8 є алгоритмом стиснення з втратами, що використовується в форматі WEBP, для зменшення розміру файлу. Він базується на використанні дискретного косинусного перетворення (DCT) [12], аналогічного до того, що використовується в JPEG. DCT перетворює зображення з просторового (піксельного) представлення до частотного. Це дозволяє видалити високочастотні компоненти, які для людського ока менш чутливі, зменшуючи розмір файлу без значного зниження якості зображення.

На відміну від JPEG, у форматі WEBP, який використовує алгоритм стиснення VP8, розбивка на блоки може бути більш гнучкою. VP8 може використовувати блоки різного розміру, включаючи 4x4, 8x8, 16x16 та інші. Це дозволяє алгоритму оптимізувати стиснення для різних частин зображення.

Також, варто зазначити, що VP8 відрізняється від JPEG ще декількома нюансами. По-перше, VP8 використовує більш ефективний метод кодування, який включає в себе прогнозування між кадрами та внутрішньо-кадрове прогнозування. По-друге, VP8 включає в себе вбудований механізм корекції помилок, що дозволяє відновити зображення навіть при втраті даних під час передачі. По-третє, VP8 підтримує альфа-канал, що дозволяє створювати прозорі зображення.

Важливо зазначити, що, хоча VP8 є алгоритмом стиснення з втратами, він може бути неефективним для стиснення деяких типів даних. Наприклад, він може бути менш ефективним для стиснення зображень з високою деталлю, оскільки він може призвести до втрати дрібних деталей під час стиснення.

Незважаючи на недоліки та переваги кожного з методів стиснення, важливо враховувати багато факторів, таких як якість зображення, розмір файлу, підтримка прозорості та інші вимоги до стиснення (див. табл. 1.3).

Таблиця 1.3 – Порівняльна таблиця методів стиснення (таблиця виконана самостійно)

Тип тиснення	PNG	TIFF	GIF	JPEG	WEBP
Алгоритм	Deflate	LZW	LZW	DCT	VP8
Стиснення	Без втрат	Без втрат	Без втрат	З втратами	З втратами
Розмір файлу	Великий	Великий	Середній	Малий	Малий
Якість зображення	Найвища	Найвища	Висока	Середня	Середня
Анімація	Ні	Ні	Так	Ні	Ні
Прозорість	Ні	Ні	Так	Ні	Так
Застосування	Зображення, де важлива якість збереження	Зображення з великим рівнем деталізації	Веб-анімація, прості графіки	Інтернет, фотографії	Веб-зображення, веб-анімація

Порівняння методів стиснення показує, що кожен метод має свої переваги та недоліки, і вибір оптимального методу залежить від конкретних вимог до якості зображення та розміру файлу.

1.3 Аналіз зберігання зображень у блокчейн сховищах

Зображення, які використовуються в сучасних інформаційних системах, мають великий обсяг даних і потребують ефективних методів зберігання та передачі. У зв'язку з цим, блокчейн сховища стають об'єктом активного дослідження для зберігання зображень, оскільки вони пропонують децентралізовану, безпечну та прозору інфраструктуру зберігання даних. У цьому дослідженні буде розглянуто найпопулярніше блокчейн сховище IPFS.

IPFS є протоколом комунікації та розподілу даних, побудованим на основі блокчейн технології. Це блокчейн сховище, яке пропонує децентралізовану архітектуру для зберігання та обміну даними [15].

При обранні IPFS для зберігання зображень у блокчейн сховищі було враховано кілька важливих чинників. IPFS вирізняється своєю децентралізованістю та масштабованістю, що робить його привабливим варіантом для зберігання даних. Ця характеристика поєднується з основними принципами блокчейн технології, забезпечуючи надійність та безпеку даних.

Архітектура IPFS базується на концепції розподіленої мережі, але з додатковим шаром блокчейн технології. Кожен вузол мережі може бути як клієнтом, що зберігає дані, так і постачальником контенту [16]. Коли клієнт запитує доступ до певного файлу за його хешем, мережа автоматично знаходить найближчий вузол, який має цей файл, та передає його клієнту через блокчейн (див. рис. 1.2).

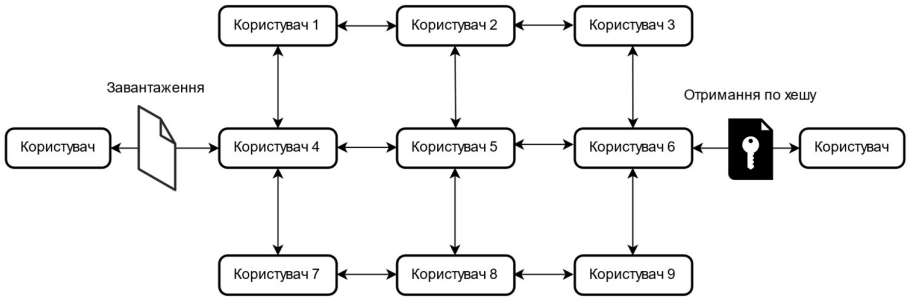

Рисунок 1.2 – Архітектура IPFS (рисунок створено самостійно)

Додатково, IPFS добре інтегрується з блокчейн технологією, що дозволяє забезпечити прозорість та цілісність збережених даних. Оскільки кожен файл отримує унікальний хеш, то він може бути легко записаний в блокчейн, забезпечуючи недоступність для змін та перевірку цілісності (див. рис. 1.3.).

Рисунок 1.3 – Взаємодія IPFS з блокчейн мережею (рисунок створено самостійно)

Враховуючи постійний розвиток IPFS та активні дослідження в галузі блокчейн технологій, обрання цього блокчейн сховища для зберігання зображень відкриває широкі перспективи для подальшого розвитку та вдосконалення цієї інфраструктури.

Незважаючи на переваги, використання IPFS як блокчейн сховища також має виклики, такі як обмежена пропускна здатність мережі, що обов'язково має бути враховано при виборі методів стиснення (див. табл. 1.4).

Таблиця 1.4 – Переваги та недоліки зберігання зображень в IPFS (таблиця виконана самостійно)

Характеристика	Плюси	Мінуси
Децентралізованість	Забезпечує високий рівень безпеки та надійності даних, оскільки дані розподіляються по різних вузлах мережі	Обмежена пропускна здатність, особливо в глобальній мережі
Масштабованість	Здатний ефективно обробляти великі обсяги даних та взаємодіяти з великою кількістю користувачів	Проблеми з пропускною здатністю при великих обсягах даних
Унікальність хешу	Кожен файл отримує унікальний хеш, що гарантує цілісність та безпеку даних	Можливість втрати доступу до даних у разі втрати хешу
Прозорість даних	Забезпечує прозорість та відкритий доступ до даних	Проблеми з конфіденційністю даних та приватністю
Вартість зберігання	Вартість зберігання даних зазвичай нижча порівняно з централізованими рішеннями	Витрати на мережеву активність та маніпуляції даними

Отже, вибір відповідних методів стиснення є ключовим для забезпечення ефективного, економічного та безпечного зберігання зображень у блокчейн сховищах.

1.4 Постановка задачі

Виходячи з усього перерахованого вище, в рамках проведення дослідження необхідно вирішити наступні вирішити наступні завдання:

- розглянути існуючи методи стиснення зображень;
- порівняти обрані для дослідження методи стиснення щоб виявити їх переваги і недоліки;
- визначити фактори, які варто враховувати при зберіганні зображень у блокчейн сховищі;
- сформувати критерії, за якими доречно порівнювати методи стиснення між собою;
- реалізувати програмну систему для проведення порівняльного аналізу;
- сформувати план проведення експерименту та визначити набір досліджуваних даних які б покривали типові сценарії використання у веб-застосунках;
- провести експеримент, задокументувати результати, провести їх порівняння та аналіз;
- надати рекомендації на основі отриманих результатів.

2 СТВОРЕННЯ ПРОГРАМНОЇ СИСТЕМИ ДЛЯ ДОСЛІДЖЕННЯ

2.1 Функціональні вимоги

Для дослідження впливу різних методів стиснення зображень на ефективність їх зберігання в блокчейні через IPFS, було розроблено веб-застосунок на основі .NET та RazorPages.

До розробленого веб-застосунку було визначено наступні функціональні вимоги:

- програма дозволяє користувачу імпортувати зображення та вибирати методи стиснення;
- зображення стискаються різними методами стиснення;
- зберігається інформація про стиснені зображення, такі як розмір файлу, якість зображення;
- стиснені зображення зберігаються в IPFS і отримується їх хеш;
- збираються показники часу завантаження в IPFS і пропускної здатності для кожного методу стиснення;
- отримані хеші зображень зберігаються в смарт контракті розгорнутому в блокчейн мережі;
- користувач може провести дослідження та отримати всі показники та їх у графічному вигляді;
- користувач може відкрити продивитись всі стисненні зображення.

Таким чином, реалізація однакових функціональних вимог за допомогою різних методів стиснення зображень дозволить провести їх порівняння без будь-яких припущень або модифікацій результатів, оскільки експеримент проводитиметься на одних і тих самих даних.

2.2 Структура проекту

Розробка програмної системи для дослідження впливу різних методів стиснення зображень на ефективність їх зберігання в блокчейні через IPFS вимагає глибокого розуміння як теоретичних аспектів, так і практичної реалізації. Основна мета даного дослідження полягає у створенні веб-застосунку, який би відповідав

усім вищезазначеним функціональним вимогам. Для цього було обрано технології та інструменти, які забезпечують високу продуктивність, безпеку та зручність у використанні.

Проект розділений на кілька основних модулів, кожен з яких відповідає за певний аспект роботи системи. Нижче наведено опис основних модулів та їх взаємодії.

2.2.1 Фронтенд частина

Фронтенд частина веб-застосунку відповідає за створення інтерфейсу користувача, який забезпечує всі необхідні дії для завантаження, стиснення та аналізу зображень. Реалізована за допомогою технології ASP.NET RazorPages, вона забезпечує інтерактивний і зручний у використанні інтерфейс. Нижче наведений детальний опис функціоналу фронтенд частини.

Користувач може завантажити одне або декілька зображень зі свого пристрою через спеціально розроблений інтерфейс завантаження. Після завантаження зображень, користувач може вибрати один або декілька методів стиснення за допомогою групи чекбоксів. Нижче наведено приклад коду, для виконання описаного функціоналу:

```
<form id="upload-form">
    <input type="file" id="image-upload" multiple>
    <div id="compression-methods">
        <label><input type="checkbox" value="jpeg"> JPEG</label>
        <label><input type="checkbox" value="png"> PNG</label>
        <label><input type="checkbox" value="webp"> WebP</label>
        <label><input type="checkbox" value="tiff"> TIFF</label>
        <label><input type="checkbox" value="gif"> GIF</label>
    </div>
    <button type="submit">Analyze</button>
</form>
```

Після завершення процесу стиснення та завантаження зображень в IPFS, користувач може переглянути всю зібрану інформацію про стиснені зображення. Це включає розмір файлу, якість зображення (метрика PSNR), час завантаження в IPFS та пропускну здатність (див. рис. 2.1).

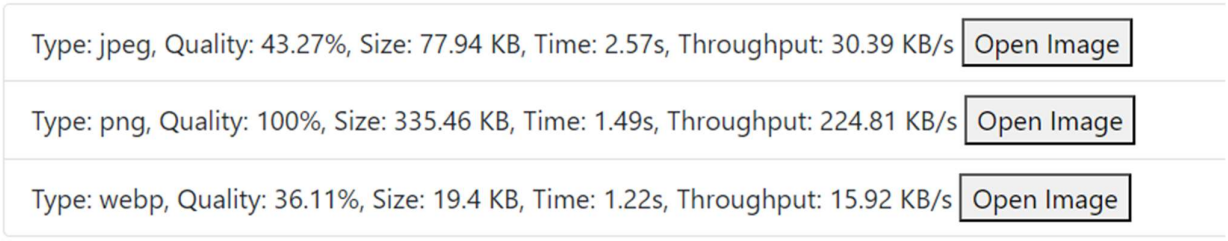

Рисунок 2.1 – Результат виведення характеристик стиснення зображення (рисунок створено самостійно)

Користувач має можливість відкрити та переглянути всі стиснені зображення для візуального порівняння якості зображень, стиснутих різними методами. Для полегшення аналізу результати представлені у графічному вигляді. Користувач може переглянути графіки, що відображають ефективність різних методів стиснення, і порівняти їх між собою. Першим відображається графік зміни розміру зображень у відсотках відносно оригінального розміру зображень для обраних методів стиснення (див. рис. 2.2).

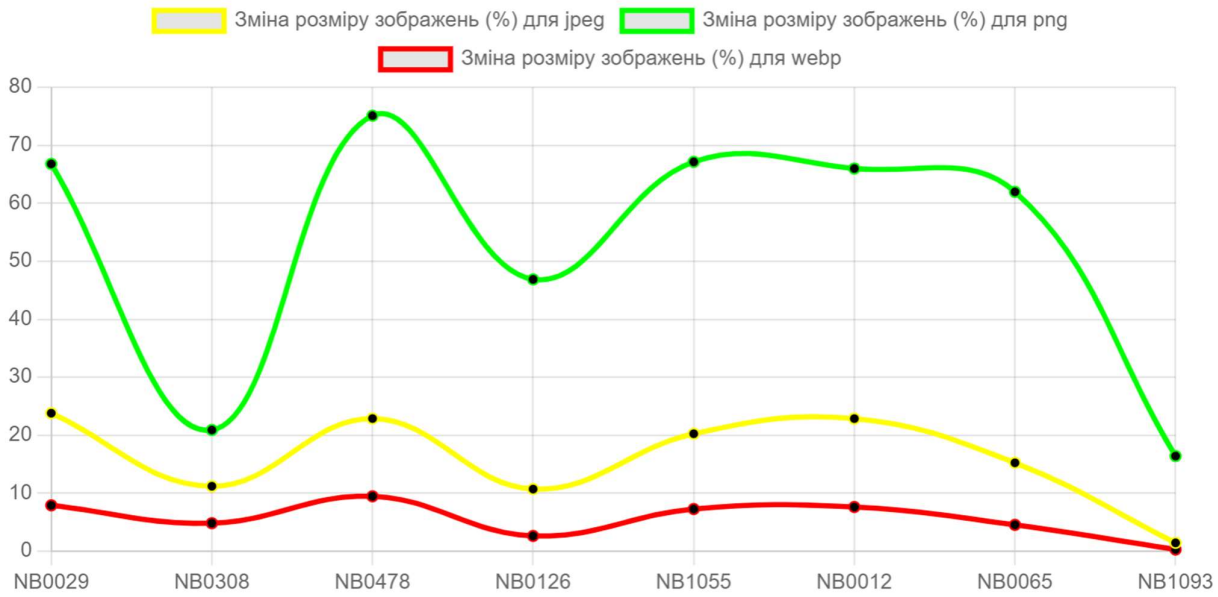

Рисунок 2.2 – Графік зміни розміру зображень (рисунок створено самостійно)

Цей графік наочно демонструє ефективність кожного методу стиснення з точки зору зменшення розміру файлу. Він дозволяє швидко порівняти, які методи

найкраще знижують розмір зображення, що є важливим для оптимізації збереження зображень в IPFS [3].

Наступним користувач може проглянути графік зміни якості стиснених зображень у відсотках відносно до якості оригінального зображення, яка визначається як 100% (див. рис. 2.3).

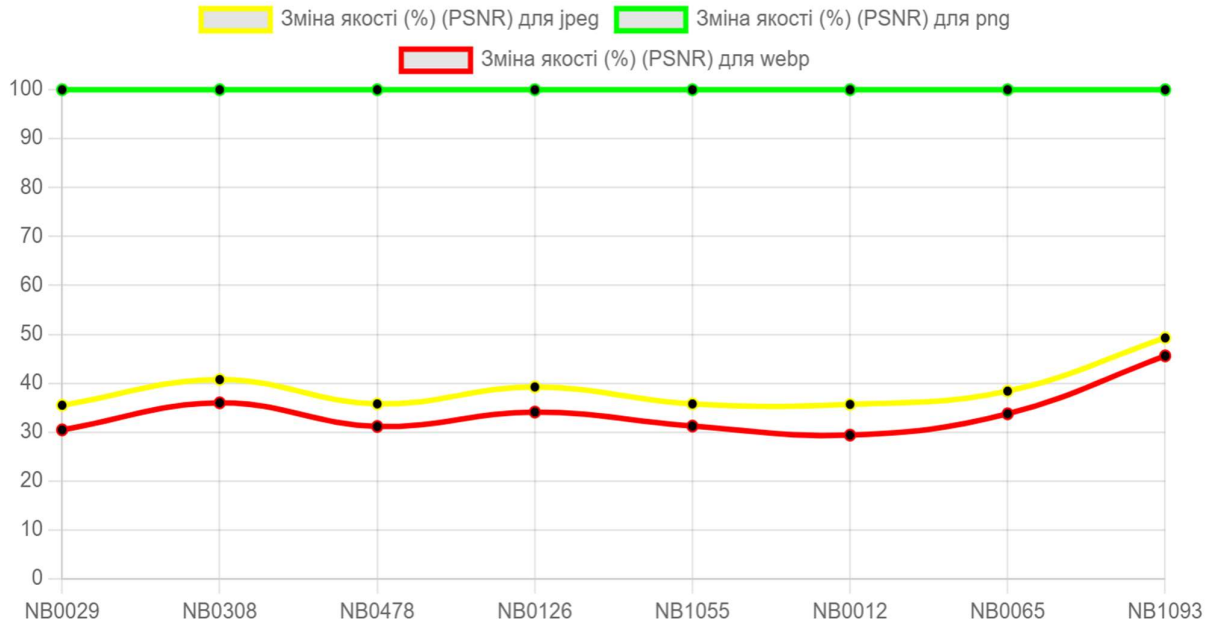

Рисунок 2.3 – Графік зміни якості зображень (рисунок створено самостійно)

Графік відображає зміну якості стиснених зображень у відсотках відносно оригіналу, що дозволяє оцінити вплив кожного методу стиснення на якість. Це допомагає знайти баланс між зменшенням розміру файлу та збереженням прийнятної якості зображення.

І останній графік, який є доступним користувачу на даному етапі дослідження, це графік витраченої пропускної здатності в кілобайтах на секунду під час збереження в IPFS для кожного методу стиснення (див. рис. 2.4).

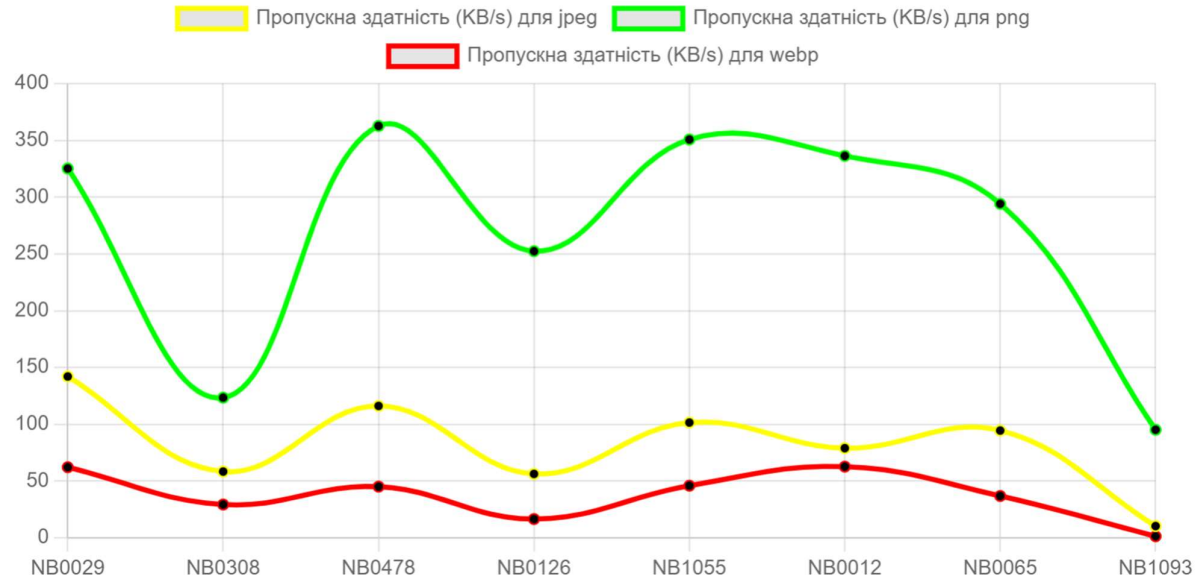

Рисунок 2.4 – Графік пропускної здатності зображень (рисунок створено самостійно)

Цей графік є корисним для оцінки ефективності мережевих ресурсів і допомагає визначити, які методи стиснення вимагатимуть менше мережевих ресурсів для завантаження файлів.

Цей підхід до розробки фронтенд частини забезпечує зручний та ефективний інтерфейс для користувача, що дозволяє легко виконувати всі необхідні дії і отримувати важливу інформацію для аналізу.

2.2.2 Бекенд частина

Бекенд частина відповідає за обробку даних, взаємодію з IPFS та блокчейном, а також за виконання основних бізнес-логік застосунку. Реалізована на основі .NET 8 та C# 12.0, вона забезпечує надійність та високу продуктивність.

Основні функції бекенд частини включають взаємодію між фронтенд та бекенд частинами через розроблені API. Це дозволяє передавати дані та виконувати необхідні операції. Нижче наведено приклад коду API частини, для завантаження зображення з фронтенд:

```
[HttpPost]
public async Task Analyze([FromForm] List<ImageData> data)
{
```

```
        await saveService.SendDatasAsync(data);
    }
```

Використовуючи бібліотеку ImageMagick, бекенд частина здійснює стиснення зображень за обраними методами, обчислюючи розмір файлу та метрику якості PSNR. Наприклад, код для стиснення зображення виглядає так:

```
    public MemoryStream CompressImage(MemoryStream imageStream, string
method)
    {
        var compressionType =
    (CompressionType)Enum.Parse(typeof(CompressionType), method, true);

        imageStream.Position = 0
        var imageBytes = imageStream.ToArray();

        var outputStream = new MemoryStream();
        switch (compressionType)
        {
            case CompressionType.Png:
            CompressPng(imageBytes, outputStream);
            break;
            ...
        }

        outputStream.Position = 0;
        return outputStream;
    }

    private void CompressPng(byte[] imageBytes, Stream outputStream)
    {
        using (var image = new MagickImage(imageBytes))
        {
            image.Format = MagickFormat.Png;
            image.Write(outputStream);
        }
    }
```

За допомогою HttpClient відбувається завантаження стиснених зображень до IPFS через сервіс Pinata та отримання унікальних хешів для кожного зображення. Код для завантаження в IPFS та ортимання хешу виглядає наступним чином:

```
    var formData = new MultipartFormDataContent();
    var streamContent = new StreamContent(compressImageStream);
    formData.Add(streamContent, "file", $"{imageName}.{method}");

    var stopwatch = Stopwatch.StartNew();
```

```
    var response = await
httpClient.PostAsync("https://api.pinata.cloud/pinning/pinFileToIPFS",
formData);
    stopwatch.Stop();

    if (response.IsSuccessStatusCode)
    {
        var responseContent = await response.Content.ReadAsStringAsync();
        var responseObject =
    JsonConvert.DeserializeObject<dynamic>(responseContent);
        var ipfsHash = (string)responseObject.IpfsHash;
        ...
    }
```

Для збору показників часу завантаження в IPFS, пропускної здатності та інших необхідних даних для аналізу ефективності методів стиснення, ми використовуємо Stopwatch для вимірювання часу і обчислюємо розмір файлу після стиснення. Код для цього процесу виглядає наступним чином:

```
var timeTaken = stopwatch.Elapsed.TotalSeconds;
var dataSize = compressedImageBytes.Length; // size in bytes
var dataSizeKb = dataSize / 1024.0; // size in KB
var sizeChange = dataSize / (originalImageSize / 100.0);
var throughput = dataSizeKb / timeTaken; // KB per second
```

Також використовується метод PSNR для оцінки якості стисненого зображення шляхом порівняння його з оригіналом. Вищий PSNR означає кращу якість стиснення. Принцип роботи алгоритму PSNR можна зрозуміти з наведеного нижче коду:

```
    private double CalculatePSNR(MagickImage originalImage, MagickImage
compressedImage)
    {
        if (originalImage.Width != compressedImage.Width ||
originalImage.Height != compressedImage.Height)
        {
            throw new ArgumentException("Images must have the same
dimensions to calculate PSNR.");
        }

        double mse = 0;
        var originalPixels = originalImage.GetPixels();
        var compressedPixels = compressedImage.GetPixels();

        for (int y = 0; y < originalImage.Height; y++)
        {
            for (int x = 0; x < originalImage.Width; x++)
```

```
            {
                var originalPixel = originalPixels.GetPixel(x,
y).ToColor();
                var compressedPixel = compressedPixels.GetPixel(x,
y).ToColor();

                mse += Math.Pow(originalPixel.R - compressedPixel.R, 2);
                mse += Math.Pow(originalPixel.G - compressedPixel.G, 2);
                mse += Math.Pow(originalPixel.B - compressedPixel.B, 2);
            }
        }

        mse /= (originalImage.Width * originalImage.Height * 3);

        if (mse == 0)
        {
            return 100;
        }

        double psnr = 10 * Math.Log10(Math.Pow(255, 2) / mse);

        return psnr;
    }
```

Метод для обчислення PSNR виконує наступні кроки:

- переконується, що обидва зображення мають однакові розміри, оскільки PSNR розраховується тільки для зображень однакових розмірів;
- ініціалізує змінну mse для зберігання значення середньоквадратичної похибки;
- проходиться по всіх пікселях зображень і обчислює різницю між значеннями кольорів оригінального та стисненого зображень для кожного каналу (R, G, B);
- підсумовуються квадрати різниць для всіх пікселів;
- ділить результат на загальну кількість пікселів і каналів;
- якщо значення MSE дорівнює нулю (що означає, що зображення ідентичні), PSNR встановлюється на максимальне значення 100;
- якщо MSE не дорівнює нулю, обчислює PSNR за формулою 2.1:

$$psnr = 10 \cdot \log_{10}\left(\frac{255^2}{mse}\right), \tag{2.1}$$

де $psnr$ – пікове відношення сигналу до шуму,

mse – значення середньоквадратичної похибки.

Використовуючи бібліотеку Nethereum, хеші зображень зберігаються у смарт контракті, розгорнутому в локальній блокчейн мережі, яка імітує Ethereum і аналогічно дістаються зі смарт контракта за допомогою Nethereum [17].

Після чого за допомогою HttpClient відбувається вивантаження стиснених зображень з IPFS за унікальним хешем, отриманим зі смарт контракта:

```
public async Task<Stream> GetDataAsync(string imageName)
{
      var cid = await blockchainStorage.GetDataAsync(imageName);
      var httpClient = new HttpClient();
      var response = await httpClient.GetAsync("https://ipfs.io/ipfs/"
+ cid);

      if (!response.IsSuccessStatusCode)
      {
            throw new Exception($"Failed to retrieve file from IPFS with
      status code: {response.StatusCode}");
      }

      var dataStream = await response.Content.ReadAsStreamAsync();

      return dataStream;
}
```

Бекенд частина є критично важливою для забезпечення інтеграції всіх компонентів системи. Вона координує обмін даними між фронтенд частиною, IPFS та блокчейном, а також забезпечує реалізацію бізнес-логіки застосунку. Завдяки використанню сучасних технологій та інструментів, таких як .NET 8, C# 12.0, ImageMagick та Nethereum, бекенд частина гарантує високу продуктивність, надійність та масштабованість системи. Це дозволяє ефективно виконувати всі необхідні операції, забезпечуючи користувачів точними й швидкими результатами дослідження.

2.2.3 Блокчейн частина

Блокчейн частина відповідає за збереження хешів зображень у децентралізованій мережі, що гарантує їх безпеку, незмінність та прозорість. Для цього використовується локальна блокчейн мережа, розгорнута за допомогою Ganache, яка імітує Ethereum блокчейн.

Локальна блокчейн мережа розгортається за допомогою Ganache, що дозволяє швидко створити середовище для тестування та розробки смарт контрактів. Ganache надає можливість створити локальну мережу Ethereum з миттєвими транзакціями, що значно пришвидшує процес розробки (див. рис. 2.5).

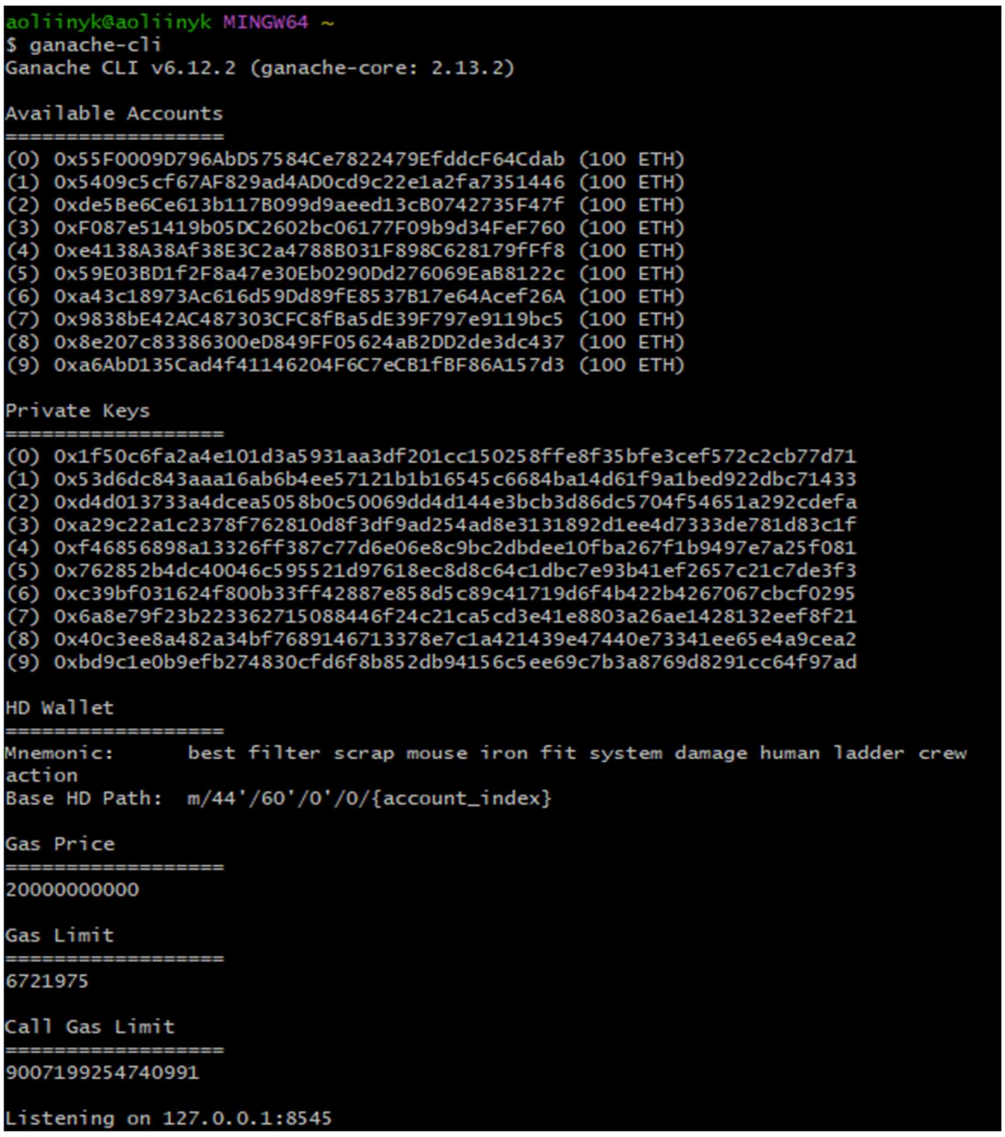

Рисунок 2.5 – Створення локальної блокчейн мережі (рисунок створено самостійно)

Смарт контракти, написані на мові Solidity, публікуються в локальну блокчейн мережу за допомогою утиліти Truffle, яка забезпечує зручний інструментарій для управління контрактами. Наприклад, код створення смарт контракту, який буде зберігати словник, де ключ – назва зображення, а значення – хеш в IPFS, може виглядати так:

```
pragma solidity ^0.5.17;

contract SimpleStorage {
  mapping(string => string) public data;

  function set(string memory _key, string memory _value) public {
    data[_key] = _value;
  }

  function get(string memory _key) public view returns (string memory)
{
    return data[_key];
  }
}
```

Далі треба зробити міграцію цього смарт контракту. Файл міграції у контексті розробки на Ethereum є скриптом, який визначає, які смарт-контракти повинні бути розгорнуті на блокчейні. Код файлу міграції наведений далі:

```
const YourContract = artifacts.require("SimpleStorage");

module.exports = function(deployer) {
    deployer.deploy(YourContract)
        .then(() => {
            console.log("Contract deployed at address:",
YourContract.address);
        });
};
```

Ключова частина цього скрипту – це виклик deployer.deploy(), який розгортає смарт контракт SimpleStorage, згаданий вище, на блокчейні. Після успішного розгортання, функція виводить у консоль повідомлення з адресом розгорнутого смарт-контракту.

Цей файл міграції використовується для того, щоб автоматизувати процес розгортання смарт-контрактів, забезпечуючи зручність та надійність в ході

розробки. Крім того, він дозволяє легко керувати та відслідковувати розгорнуті контракти.

Truffle є одним з найпоширеніших фреймворків для розробки смарт контрактів на Ethereum, оскільки він спрощує процес компіляції, тестування та розгортання смарт контрактів.

Смарт контракти написані на мові Solidity публікуються в локальну блокчейн мережу за допомогою утиліти Truffle, яка забезпечує зручний інструментарій для управління контрактами (див. рис. 2.6).

```
aoliinyk@aoliinyk MINGW64 ~/RiderProjects/InvestigateStoringImagesInBlockchainWe
bApp/MyContract (master)
$ truffle migrate --reset

Compiling your contracts...
===========================
> Compiling .\contracts\SimpleStorage.sol
> Artifacts written to C:\Users\aoliinyk\RiderProjects\InvestigateStoringImagesI
nBlockchainWebApp\MyContract\build\contracts
> Compiled successfully using:
   - solc: 0.5.17+commit.d19bba13.Emscripten.clang


Starting migrations...
======================
> Network name:    'development'
> Network id:      1717876403805
> Block gas limit: 6721975 (0x6691b7)


2_deploy_contracts.js
=====================

   Replacing 'SimpleStorage'
   -------------------------
   > transaction hash:    0x7d3eb0e6f1c41be7ad7fe5a83a55c03aace1e99b6da87af9a116
c4783417a3a8
- Blocks: 0            Seconds: 0
   > Blocks: 0            Seconds: 0
   > contract address:    0x4F0C74dc7C59C0Ce4e5Ae87Ca53AB0cAE3f709B2
   > block number:        155
   > block timestamp:     1717885689
   > account:             0x55F0009D796AbD57584Ce7822479EfddcF64Cdab
   > balance:             99.53543172
   > gas used:            449557 (0x6dc15)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00899114 ETH

Contract deployed at address: 0x4F0C74dc7C59C0Ce4e5Ae87Ca53AB0cAE3f709B2
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00899114 ETH

Summary
=======
> Total deployments:   1
> Final cost:          0.00899114 ETH
```

Рисунок 2.6 – Публікація смарт контракта в локальну блокчейн мережу за допомогою Truffle (рисунок створено самостійно)

Після того як смарт контракт був успішно розгорнутий в локальній мережі, можна використовувати отримані за результатами розгортання адресу смарт контракту і приватний ключ для збереження хешу зображення в блокчейні. Код для збереження хешу в смарт контракті:

```
public async Task<string> SaveImageHashAsync(string imageName, string
ipfsHash)
{
    var contract = _web3.Eth.GetContract(contractAbi,
contractAddress);
    var function = contract.GetFunction("set");
    var functionData = function.GetData(imageName, ipfsHash);

    var nonce = await
_web3.Eth.Transactions.GetTransactionCount.SendRequestAsync(account.A
ddress, BlockParameter.CreatePending());

    var transactionInput = new TransactionInput
    {
        From = _account.Address,
        To = _contractAddress,
        Nonce = nonce,
        Data = functionData
    };

    var offlineTransactionSigner = new
AccountOfflineTransactionSigner();
    var signedTransaction =
offlineTransactionSigner.SignTransaction(account, transactionInput);

    var transactionHash = await
_web3.Eth.Transactions.SendRawTransaction.SendRequestAsync(signedTran
saction);

    return transactionHash;
}
```

Так само хеші зображень дістаються з смарт контракта за допомогою Nethereum, для цього використовується наступний код:

```
public async Task<string> GetDataAsync(string imageName)
{
    var contract = _web3.Eth.GetContract(_contractAbi,
_contractAddress);
    var function = contract.GetFunction("get");
    return await function.CallAsync<string>(imageName);
}
```

Завдяки інтеграції блокчейн технологій, система забезпечує високий рівень безпеки та надійності зберігання даних [18]. Використання смарт контрактів дозволяє автоматизувати процес зберігання та доступу до хешів зображень, що підвищує ефективність та прозорість операцій [19]. Локальна блокчейн мережа, розгорнута за допомогою Ganache, забезпечує гнучке та зручне середовище для розробки та тестування, що сприяє швидкому впровадженню та перевірці нових функціональних можливостей системи. Таким чином, блокчейн частина є невід'ємним компонентом, який гарантує незмінність та доступність даних, що є критично важливим для децентралізованих систем зберігання інформації .

Нижче наведено малюнок, який візуалізує взаємодію вищезазначених модулів розробленої системи (див. рис. 2.7).

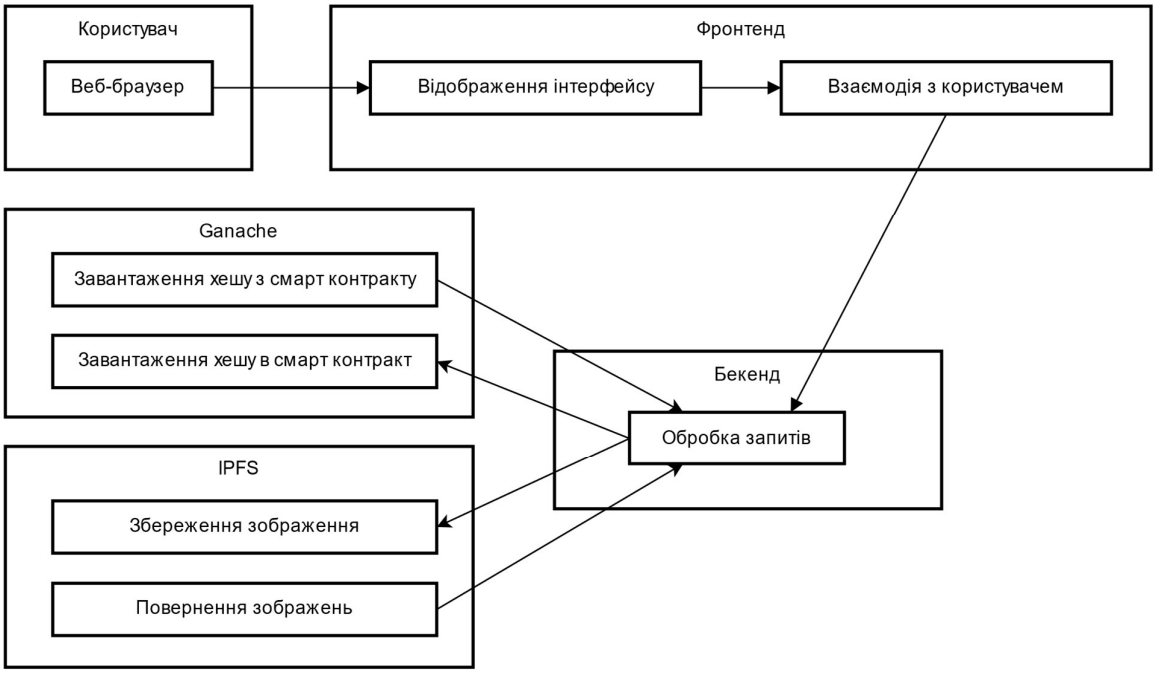

Рисунок 2.7 – Взаємодія модулів розробленої системи (рисунок створено самостійно)

У цій візуалізації показано, як різні компоненти системи взаємодіють один з одним для забезпечення ефективного зберігання, стиснення та передачі зображень, що дозволяє надійно зберігати великі обсяги даних.

3 ОПИС ПРОВЕДЕНИХ ДОСЛІДЖЕНЬ

Для дослідження ефективності різних методів стиснення зображень у контексті їх зберігання та передачі в IPFS було обрано різноманітний набір зображень. Зображення представляють типові сценарії використання у веб-застосунках і включають фотографії, медичні зображення, графіку та логотипи. Всі зображення були отримані у форматі BMP, що забезпечує високу якість без втрат і є оптимальним для подальшого порівняльного аналізу методів стиснення.

Для кожного зображення було застосовано п'ять популярних методів стиснення:

- JPEG;
- PNG;
- WEBP;
- GIF;
- TIFF.

Ці формати були обрані через їх широке використання у веб-застосунках та різні підходи до стиснення і збереження якості зображень. JPEG використовується для фотографій завдяки своєму ефективному алгоритму стиснення з втратами, який добре працює з деталями та градієнтами кольору. PNG забезпечує стиснення без втрат, що робить його ідеальним для графіки та ілюстрацій. WEBP поєднує в собі переваги як стиснення з втратами, так і без втрат, пропонуючи високий ступінь компресії. GIF часто використовується для простих графічних зображень і анімацій, тоді як TIFF забезпечує високоякісне стиснення без втрат для збереження максимальної якості.

Набір досліджуваних даних включає в себе 20 фотографій, які можна відрізняються наступними характеристиками:

- наявність монохромності;
- кольорова гамма та теплота;
- контрастність;
- насиченість;

– різна щільність пікселів.

Також набір даних містить 15 логотипів, які можна розділити на 3 групи по 5 логотипів:

– кольорові логотипи;
– чорно-білі логотипи;
– чорно-білі логотипи без заливки кольором.

Окрім цього датасет включає в себе 5 графічних зображень, серед яких є чорно-білі зображення. Графічні зображення відрізняються за розміром, кольором, насиченістю та іншими характеристиками.

Також для всебічного аналізу, в досліджувані дані було додано чорно-білі медичні зображення, такі як знімки МРТ, зображення клітин та інші зображення з медичної сфери.

Кожне зображення з набору даних було ретельно підібране, щоб включати широкий спектр характеристик, таких як:

– роздільна здатність, тобто кількість пікселів по горизонталі та вертикалі, яка впливає на розмір файлу та якість зображення;
– щільність пікселів, що визначається кількістю пікселів на дюйм і впливає на чіткість зображення;
– кольорова гамма, яка відображає діапазон кольорів, що наявні на зображенні;
– контрастність, тобто різниця між найтемнішими і найсвітлішими частинами зображення;
– насиченість, яка ілюструє інтенсивність кольорів на зображенні.

Вибір вище-зазначених характеристик дозволяє максимально точно оцінити ефективність кожного методу стиснення для різних типів зображень.

Після того як був сформований набір даних, стало можливим провести аналіз досліджуваних зображень, для того щоб визначити оптимальний метод стиснення для зберігання у блокчейн сховищі. Нижче описуються показники, що використовуються при аналізі.

Оскільки одною з основних цілей під час зберігання файлів є зниження витрат на зберігання [20], то було визначено один з головних показників, який прямо впливає на вартість зберігання зображень в IPFS – розмір завантаженого файлу. Відсоткове співвідношення розміру стисненого файлу до початкового файлу обчислювалось за формулою 3.1:

$$S = \frac{C}{O} \cdot 100, \tag{3.1}$$

де S – відсоткове співвідношення розмірів,

C – розмір стиснутого файлу,

O – розмір оригінального файлу.

Аналогічно до розміру файлу, на вартість зберігання також впливає витрачена пропускна здатність каналу (мережі) який був наданий кінцевому користувачу. Пропускна здатність, вимірювана у кілобайтах на секунду обчислювалась за формулою 3.2:

$$Tp = \frac{C}{t}, \tag{3.2}$$

де Tp – пропускна здатність,

C – розмір стиснутого файлу у кілобайтах,

t – час, витрачений на завантаження в IPFS, вимірюваний в секундах.

Також, чи не найважливішою вимогою для клієнта, під час процесу стиснення є оптимальна компресія [20]. Тобто максимально можливе збереження якості файлу після його форматування. Якість файлу, яка вимірюється шляхом калькуляції піксельної різниці сигналу до шуму (PSNR), розраховувалась за формулою 3.3:

$$psnr = 10 \cdot \log_{10}\left(\frac{255^2}{mse}\right) \quad (3.3)$$

де $psnr$ – пікове відношення сигналу до шуму,

mse – значення середньоквадратичної похибки.

Ці показники були обрані для об'єктивної оцінки ефективності кожного методу стиснення з точки зору розміру файлу, швидкості завантаження, та якості зображення.

Ретельно підібраний набір даних та застосовані методи аналізу дозволяють зробити обґрунтовані висновки щодо ефективності різних методів стиснення зображень. Це важливо для оптимізації роботи децентралізованих систем зберігання даних, таких як IPFS, та покращення продуктивності веб-застосунків.

4 АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ

У цьому розділі проведено аналіз результатів досліджень методів стиснення зображень для зберігання в блокчейн сховищах на прикладі IPFS. Було досліджено кілька ключових показників для різних категорій зображень: фотографії, медичні зображення, графіка та логотипи. Це дозволяє оцінити ефективність кожного методу стиснення для конкретного типу зображень.

В першу чергу, було виміряно та проаналізовано розміри фотографій після стиснення за допомогою різних методів (див. рис. 4.1).

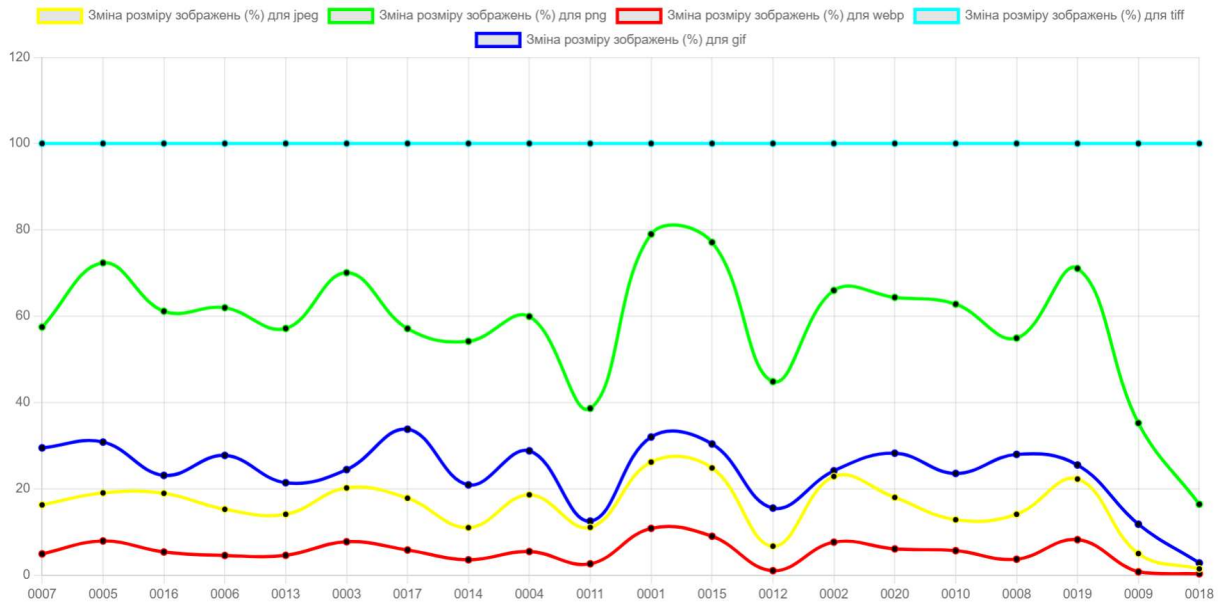

Рисунок 4.1 – Графік зміни розміру фото (рисунок створено самостійно)

Як видно з діаграми, найбільший розмір файлу після стиснення спостерігався для методу TIFF (алгоритм LZW). Це можна пояснити тим, що методи стиснення без втрат зазвичай зберігають більше інформації, ніж методи з втратами. З іншого боку, метод WEBP (алгоритм VP8) показав найкращий результат з точки зору зменшення розміру файлу.

Другим важливим показником було вимірювання якості зображення після стиснення за допомогою метрики PSNR (Peak Signal-to-Noise Ratio) (див. рис. 4.2).

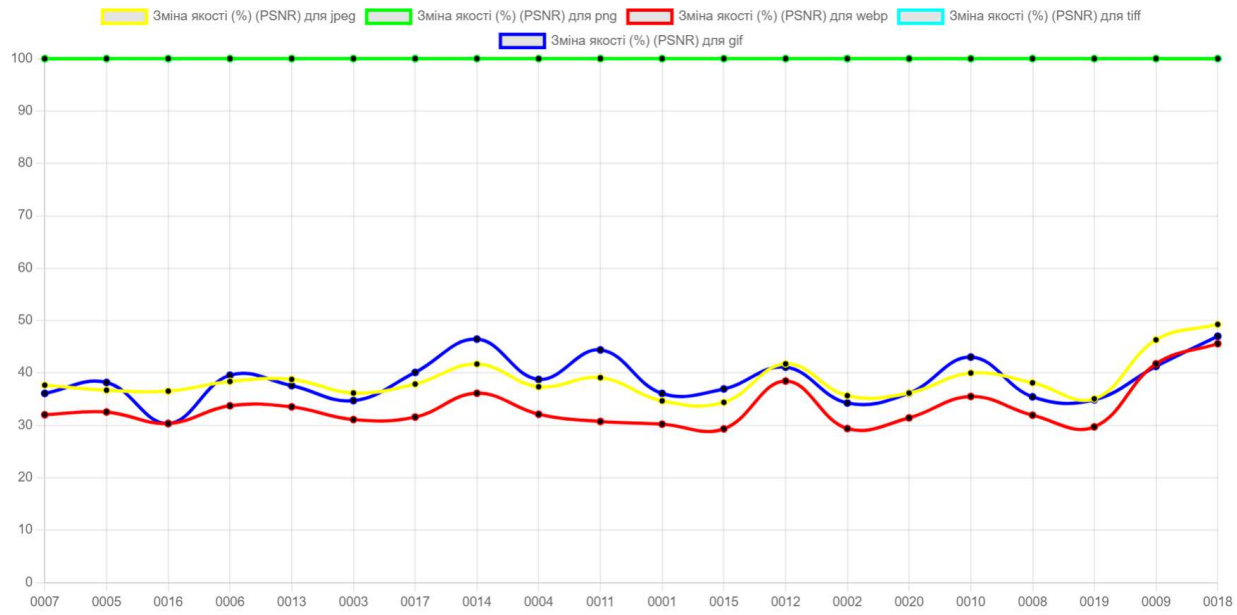

Рисунок 4.2 – Графік зміни якості фото (рисунок створено самостійно)

Як показує діаграма, методи стиснення без втрат, такі як PNG і TIFF, повністю зберігають якість зображення, що підтверджується максимальними значеннями PSNR. Методи стиснення з втратами, такі як JPEG, WEBP і GIF, демонструють нижчі значення PSNR, що є очікуваним результатом через втрату частини інформації при стисненні.

Наступним важливим показником була виміряна пропускна здатність при завантаженні зображень в IPFS (див. рис. 4.3).

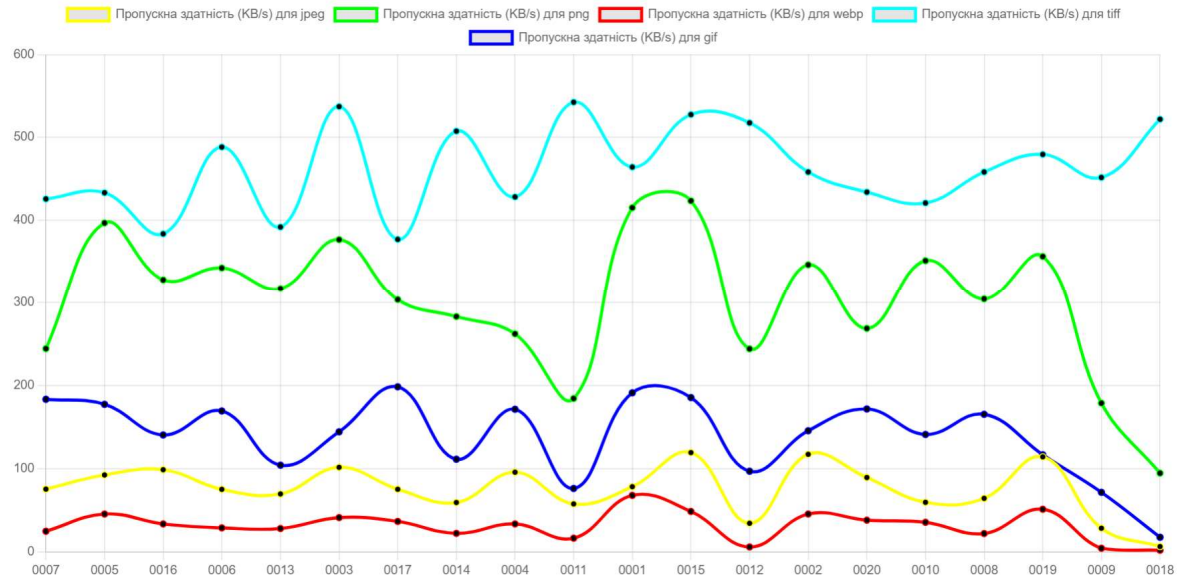

Рисунок 4.3 – Графік пропускної здатності фото (рисунок створено самостійно)

Як показує діаграма, найбільш ефективні з точки зору пропускної здатності, тобто витрачають її менше, є методи стиснення з втратами, такі як WEBP, JPEG та GIF. Це можна пояснити тим, що менші розміри файлів дозволяють швидше передавати дані через мережу.

Аналіз показав, що вибір методу стиснення залежить від конкретних вимог до програми. Для фотографій, які зазвичай містять велику кількість деталей та кольорів, методи стиснення з втратами, такі як JPEG і WEBP, є більш ефективними для зменшення розміру файлу. Однак для програм, де критично важлива висока якість фотографій, методи стиснення PNG є більш придатними. На основі результатів ми робимо висновок, що метод JPEG забезпечує оптимальний баланс між зменшенням розміру файлу та якістю зображення для стиснення фотографій.

Аналогічно фотографіям, було виміряно та проаналізовано розміри медичних зображень після стиснення за допомогою різних методів (див. рис. 4.4).

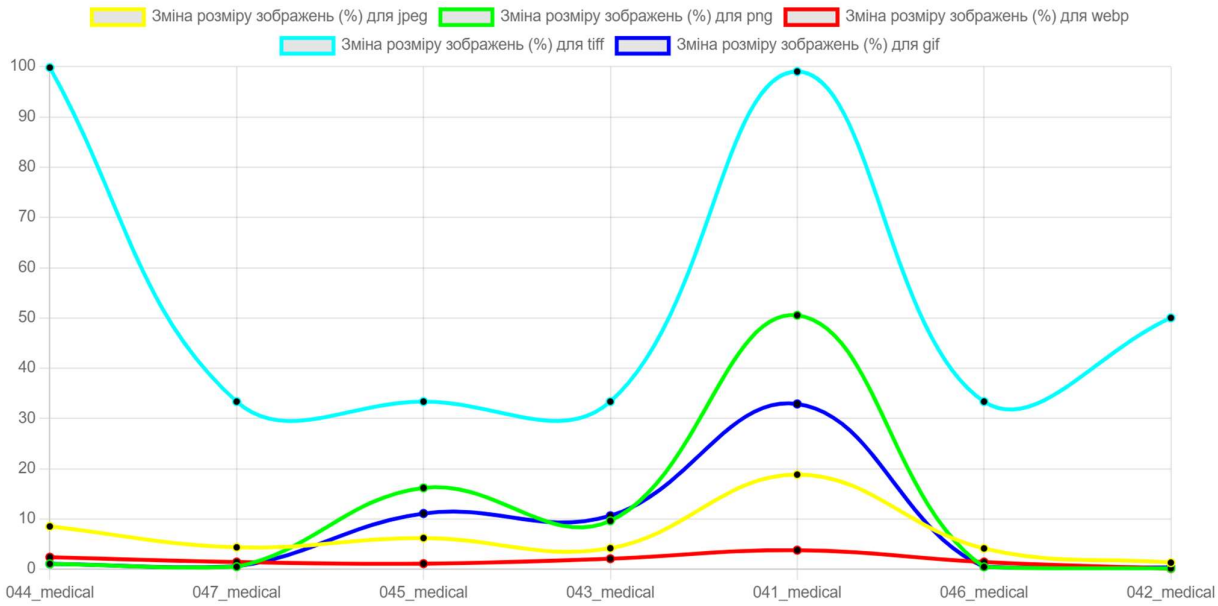

Рисунок 4.4 – Графік зміни розміру медичних зображень (рисунок створено самостійно)

Як видно з діаграми, найбільший розмір файлу після стиснення спостерігався для методу TIFF. Методи стиснення з втратами, такі як JPEG та WEBP, значно зменшили розмір файлу, пропонуючи ефективніші результати з точки зору обсягу даних. Також можна побачити що метод PNG, який стискає зображення без втрат,

показав найкращі результати для більшості даних, в порівняні з іншими методами стиснення, навіть з тими, що нехтують якістю під час форматування.

Другим важливим показником було вимірювання якості медичних зображень після стиснення за допомогою метрики PSNR (див. рис. 4.5).

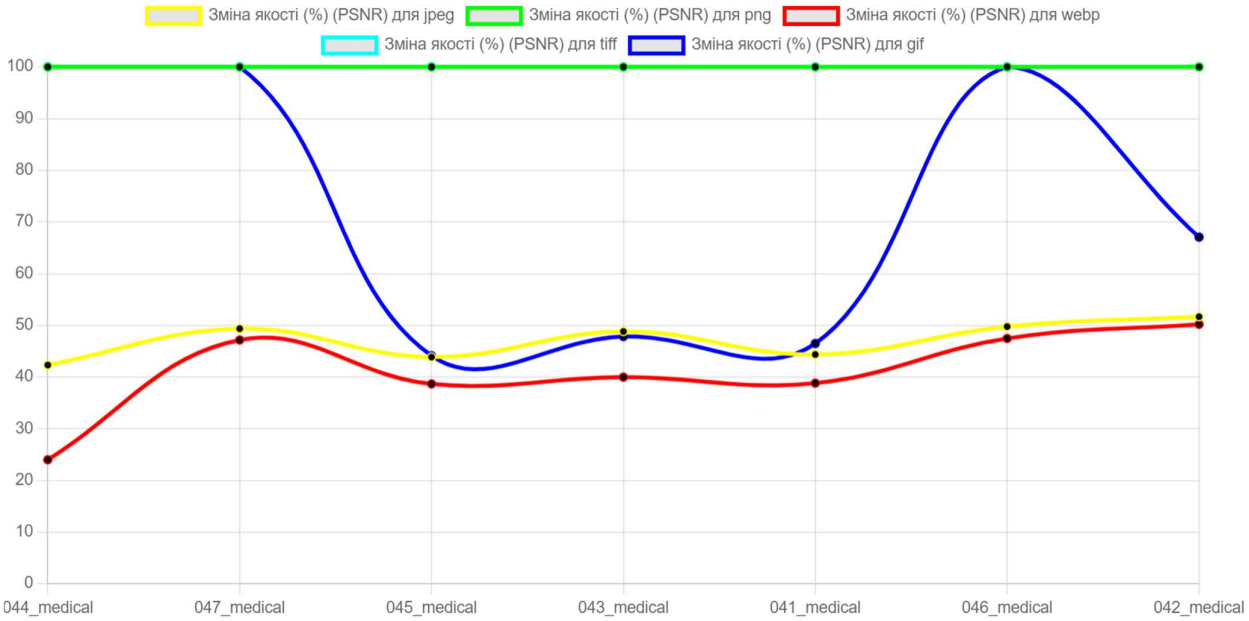

Рисунок 4.5 – Графік зміни якості медичних зображень (рисунок створено самостійно)

Як показує діаграма, методи стиснення без втрат, такі як PNG і TIFF, повністю зберігають якість зображення. Методи стиснення з втратами, такі як JPEG, WEBP та для більшості вибірки GIF, демонструють нижчі значення PSNR, що може бути критичним для медичних зображень, де важливе збереження максимальної кількості деталей.

Наступним важливим показником була виміряна пропускна здатність при завантаженні медичних зображень в IPFS (див. рис. 4.6).

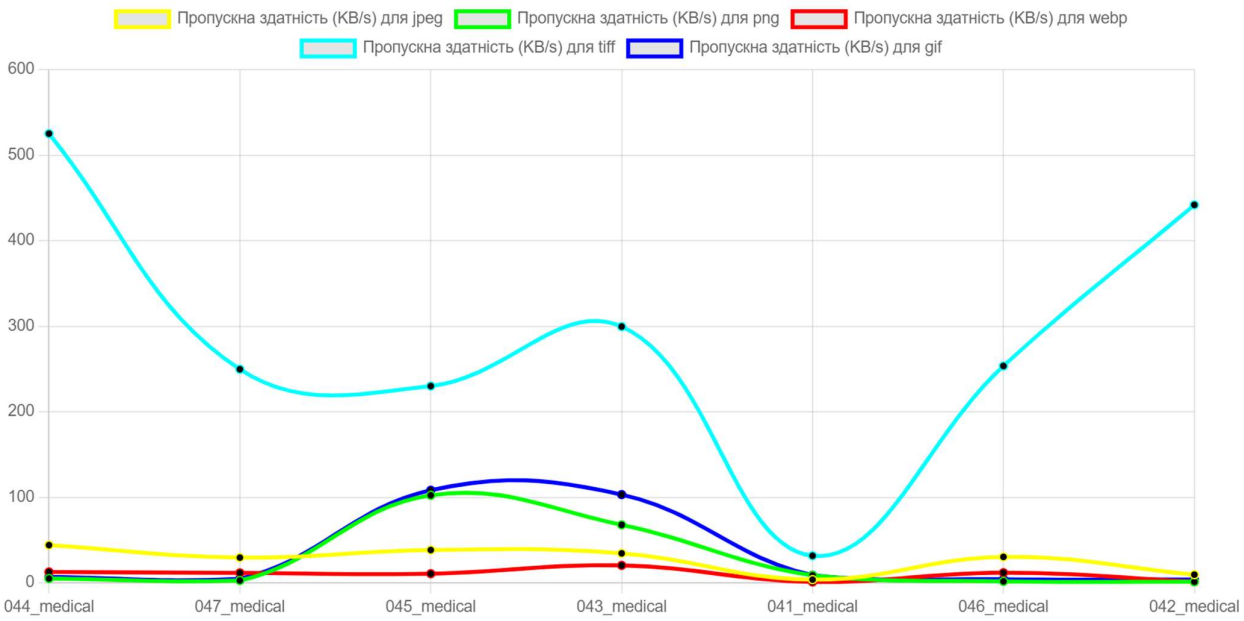

Рисунок 4.6 – Графік пропускної здатності для медичних зображень (рисунок створено самостійно)

Як показує діаграма, методи стиснення з втратами, такі як WEBP та JPEG , є найбільш ефективними з точки зору пропускної здатності, але в той же час, PNG показав хоч і не для всієї вибірки, але для більшості, найкращі показники.

Аналіз показав, що для медичних зображень критично важливим є збереження максимальної якості, тому метод PNG є більш придатним за інші, оскільки стискає зображення без втрати якості і показує найкращі результаті для більшості досліджуваних файлів за такими характеристиками як розмір файлу та пропускна здатність під час завантаження в блокчейн сховище. Проте, якщо необхідно надійно зменшити розмір файлу в декілька разів для усієї вибірки, враховуючи можливу втрату важливих деталей, може бути використаний метод стиснення з втратами WEBP, оскільки він показав найкращі результати для цього показника.

Далі було виміряно та проаналізовано розміри графіки та логотипів після стиснення за допомогою різних методів (див. рис. 4.7).

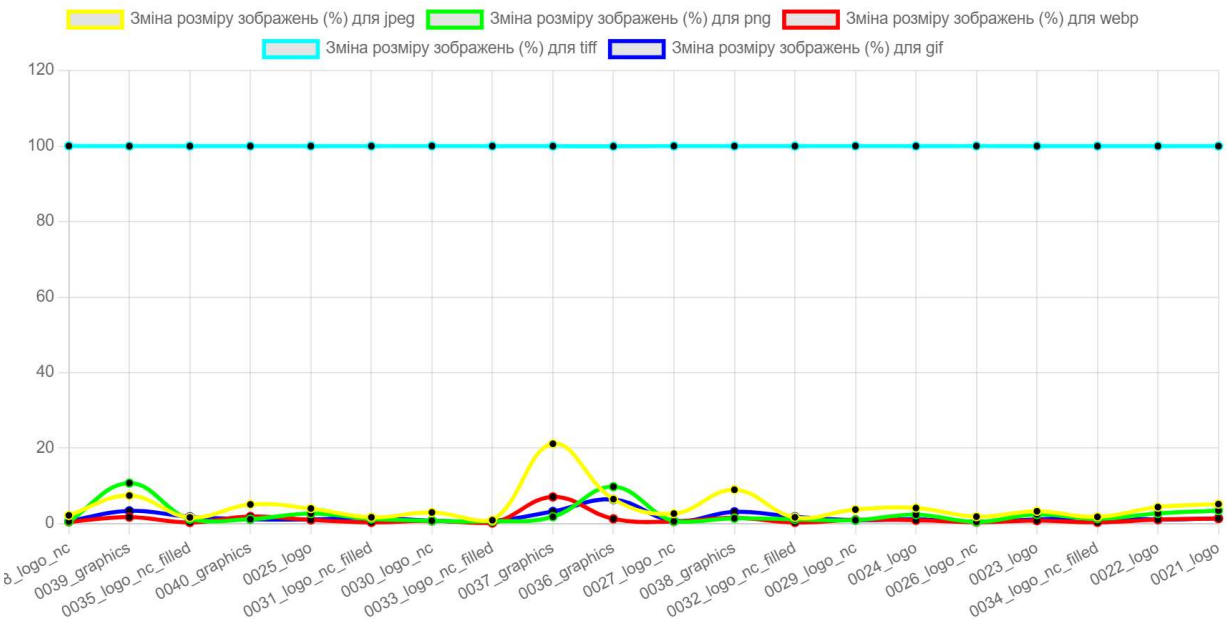

Рисунок 4.7 – Графік зміни розміру графіки і логотипів (рисунок створено самостійно)

Як видно з діаграми, метод стиснення без втрат TIFF, має більші розміри файлів у порівнянні з методами з втратами, такими як JPEG, WEBP та GIF. У той же час, метод стиснення без втрат PNG показує кращі результати стиснення за JPEG та GIF (див. рис. 4.8.).

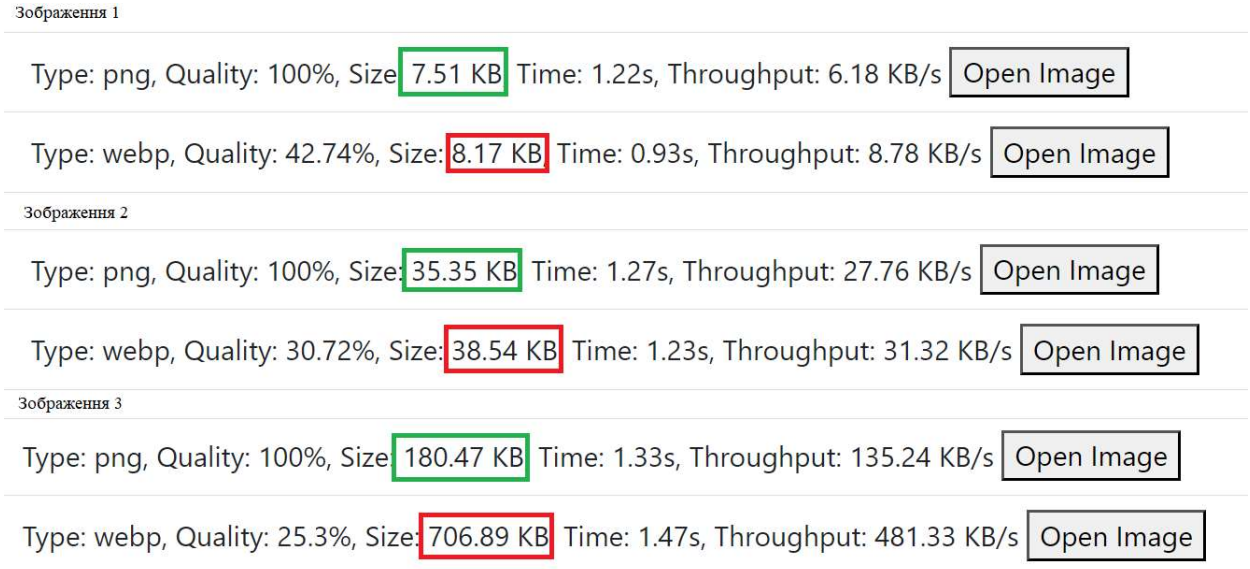

Рисунок 4.8 – Порівняння розмірів деяких файлів для WEBP та PNG (рисунок створено самостійно)

Попередній аналіз показав, що пропускна здатність прямо залежить від розміру файлу, тому проводити додатковий аналіз діаграми пропускної здатності для графічних зображень та логотипів не є необхідним. Також, оскільки для цих зображень є доречним максимальне збереження, то метод PNG є оптимальнішим за інші, оскільки проводить безвтратне стискання і показує гарні результати стиснення файлу. Проте, як альтернативу для методу PNG при збереженні графічних файлів, можна відмітити WEBP, оскільки цей метод стиснення показує чи найкращі показники співвідношення якості та розміру файлу серед методів стиснення з втратами. Проте в такому випадку буде мати місце значна втрата якості.

ВИСНОВКИ

В ході виконання даної кваліфікаційної роботи було досліджено методи стиснення зображень для зберігання у блокчейн сховищах на прикладі IPFS. Для цього були виконані наступні дії:

- розглянуто особливості зберігання даних у блокчейн сховищах, зокрема в IPFS;
- проаналізовано методи стиснення зображень, які можуть бути застосовані для зменшення розміру файлів перед зберіганням в IPFS;
- визначено показники ефективності для кожного з методів стиснення (розмір файлу, якість зображення, пропускна здатність при завантаженні, час завантаження);
- розроблено бекенд на .NET 8 та фронтенд на Razor Pages для імпорту та стиснення зображень;
- розгорнуто локальну блокчейн мережу Ethereum за допомогою Ganache
- написано смарт контракт на мові Solidity та проведений деплой в локальну блокчейн мережу за допомогою Truffle
- проведено експеримент зі стискання зображень різними методами та збереженням їх в IPFS із фіксацією отриманих хешів у смарт-контрактах на локальній блокчейн мережі;
- зібрано та проаналізовано дані про розмір файлів, якість зображень (за метрикою PSNR), час завантаження в IPFS та пропускну здатність для кожного методу стиснення;
- візуалізовано результати дослідження у вигляді графіків та таблиць.

За результатами дослідження, по-перше, було визначено, що методи стиснення без втрат, такі як PNG і TIFF, дозволяють зберегти високу якість зображеннь. Проте, це призводить до значного збільшення розміру файлу. Така особливість може бути критично важливою для типів зображень, де максимальна деталізація є необхідною, як-от медичні зображення.

По-друге, дослідження показало, що методи стиснення з втратами, такі як JPEG та WEBP, дозволяють значно зменшити розмір файлу. Це покращує ефективність зберігання та передачі даних у блокчейн сховищах. Зокрема, вони є оптимальними для використання з фотографіями та графікою, де допускається деяке зниження якості без значних втрат візуальної інформації.

Крім того, було встановлено, що метод WEBP демонструє найкращі результати з точки зору зменшення розміру файлу при збереженні достатньої якості зображення. Це робить його оптимальним для використання у випадках, коли необхідно знайти баланс між розміром файлу та якістю.

Дослідження також виявило, що пропускна здатність при завантаженні в IPFS є найвищою у методів стиснення з втратами, що зумовлено їх меншим розміром файлів. Це сприяє швидшій передачі даних у мережі, що є важливим показником ефективності.

На основі отриманих результатів можна зробити висновок, що вибір методу стиснення залежить від конкретних вимог до програми. Для випадків, де критично важлива висока якість зображень, найкращим вибором є метод стиснення без втрат PNG. Для більш ефективного використання ресурсів та зменшення часу завантаження, методи стиснення з втратами, такі як JPEG та WEBP, є більш доцільними. Проведене дослідження показало, що метод JPEG забезпечує оптимальний баланс між зменшенням розміру файлу та якістю зображення.

Для підтвердження отриманих результатів та зроблених висновків необхідні подальші дослідження. Вони мають охоплювати різноманітні типи зображень та включати більш складні сценарії використання, що дозволить отримати більш точні дані щодо ефективності кожного методу стиснення у різних умовах.

Результати даної кваліфікаційної роботи було опубліковано в журналі «Біоніка інтелекту», який включено до Переліку наукових фахових видань України, категорія «Б», технічні науки, затверджено наказом МОНУ від 02.07.2020 р. № 886 (дивись Додаток В).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Blockchain Technology Explained: Powering Bitcoin. URL: https://www.ibm.com/topics/what-is-blockchain (date of access: 29.03.2023).

2. Кравченко П. Блокчейн і децентралізовані системи: навчальний посібник для студентів закладів вищої освіти : у 3 частинах. Ч. 1 / П. Кравченко, Б. Скрябін, О. Дубініна. – Харків : ПРОМАРТ, 2018. – 400 с. – ISBN 978-617-7634-26-2.

3. The History of Blockchain Technology. URL: https://101blockchains.com/history-ofblockchaintimeline (date of access: 01.04.2023)

4. Терещенко Г. Ю., Груздо І. В. Застосування симетричних алгоритмів в блокчейні. Біоніка інтелекту. – Харків : ХНУРЕ. – 2020. – № 1 (94). – С. 33-39.

5. J. Solem. Programming Computer Vision with Python: Tools and algorithms for analyzing images. O'Reilly Media, Inc., 2012. 264 p.

6. N. Bilous, G. Tereshchenko Copyright protection using blockchain. Біоніка інтелекту – Харків : ХНУРЕ. – 2019. – № 1 (92). – С. 52-58.

7. Cryptography Hash functions. URL: https://www.tutorialspoint.com/cryptography/cryptography_hash_functions.html (date of access: 05.04.2023).

8. Білоконенко В. М , Ревенчук І. А. Алгоритми сегментації зображень на базі побудови матриць збігів. Східно-Європейський журнал передових технологій.- №2(62), том 2, 2013, с.43-45.

9. Специфікація алгоритму стискання даних формату DEFLATE версія 1.3 (RFC1951). URL : http://www.kytok.org.ua/post/cpetsyfikatsiya-alhorytmu-styskannya-danykh-formatu-deflate-versiya-1-3-rfc1951 (date of access: 06.04.2023).

10. LZW compression. URL: https://www.techtarget.com/whatis/definition/LZW-compression (date of access: 06.04.2023).

11. GIF File Format Summary. URL: https://www.fileformat.info/format/gif/egff.htm (date of access: 07.04.2023).

12. W. Pennebaker, J. Mitchell, JPEG: Still Image Data Compression Standard. – Springer New York, NY, 1992. – 638p. – ISBN: 978-0-442-01272-4.

13. JPEG YCbCr Support. URL: https://learn.microsoft.com/en-us/windows/win32/wic/jpeg-ycbcr-support (date of access: 07.04.2023).

14. Lossy Data Compression: JPEG URL: https://cs.stanford.edu/people/eroberts/courses/soco/projects/data-compression/lossy/jpeg/coeff.htm (date of access: 07.04.2023).

15. Huckle S. Internet of Things, Blockchain and Shared Economy Applications / S. Huckle, R. Bhattacharya, M. White, N. Beloff // Procedia Comput. Science. – Oct. 2016. – Vol. 98. – P. 461–466.

16. Smart Contracts: The Ultimate Guide for the Beginners. URL: https://101blockchains.com/smartcontracts (date of access: 10.04.2023).

17. What is Nethereum? – Nethereum Documentation. URL: https://docs.nethereum.com/en/latest/ (date of access: 11.04.2023).

18. Blockchain Architecture Basics: Components, Structure, Benefits & Creation. URL: https://mlsdev.com/blog/156-how-to-build-your-own-blockchain-architecture (date of access: 05.04.2023).

19. Шаронова Н.В., Терещенко Г.Ю. Проблеми і перспективи практичного застосування інформаційної технології blockchain в smart-контрактах. Інтелектуальні системи та інформаційні технологі (ISIT-2019). – Матеріали Міжн. Наук.-практ. Конф. – Одеса, 19 − 24 серпня 2019 р. – С. 214–219.

20. Аналіз методів стиснення зображень для зберігання у децентралізованих блокчейн сховищах / І. Кириченко, Г. Терещенко, А. Олійник// Біоніка інтелекту. – Харків: ХНУРЕ. – 2024. – № 1 (100). – С. 28-36.

21. Методичні вказівки до виконання кваліфікаційної роботи магістра за спеціальністю 121 – Інженерія програмного забезпечення (освітньо-наукова програма – Інженерія програмного забезпечення) для усіх форм навчання /Упор.: З. В. Дудар, В. В. Голян, В. І Каук, В. Ю. Нечволод, І. А. Ревенчук – Харків: ХНУРЕ, 2022 – 81с.