Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

на тему:

Розробка програми приховування інформації у контурах об'єктів растрових зображень

Виконав: студент 4 курсу, гр. 1КІТС-20б
спеціальності 125 – Кібербезпека
Освітня програма – Кібербезпека
інформаційних технологій та систем

_____ Бондарчук А.О._____

Керівник: к.т.н., доц. каф. МБІС
_____ Адлер О.О._____

«_6_» _червня_ 2024 р.

Рецензент: к.т.н., доцент каф. ОТ
_____Кожем'яко А.В._____

«_6_» _червня_ 2024 р.

Допущено до захисту
Голова секції УБ кафедри МБІС
д.т.н., проф. Яремчук Ю.Є.
«_6_» _червня_ 2024 р.

Вінниця ВНТУ – 2024

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти І-й (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС

_____ **д.т.н., проф. Яремчук Ю.Є.**
“ 22 ” березня 2024 р.

З А В Д А Н Н Я
на бакалаврську дипломну роботу студенту
Бондарчука Андрія Олеговича
(прізвище, ім`я, по-батькові)

1. Тема роботи Розробка програми приховування інформації у контурах об’єктів растрових зображень
Керівник роботи к.т.н., доцент кафедри МБІС Адлер Оксана Олександрівна
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від “ 11 ” березня 2024 року № 80
2. Термін подання студентом роботи
3. Вихідні дані до роботи Електронні джерела та наукові статті по темі. Програмне забезпечення, яке стосується теми бакалаврської дипломної роботи.
4. Зміст текстової частини:
Теоретичний аналіз предметної області. Проектування програмного додатку. Програмна реалізація.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов’язкових креслень)
Знаходження контуру. Пікселі як одиниця графіки. Бітова глибина і її параметри. Растрові зображення. Алгоритм роботи програми. Структура шаблону проектування. Авторизація користувача. Реєстрація користувача. Файли користувача. Форма для роботи PNG форматом. Форма для роботи JPG форматом. Таблиця «images». Результати тестування. Вибір методу. Заповнені поля. Результат приховування.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ I	Адлер О.О., к.т.н., доц. каф. МБІС		
Розділ II	Адлер О.О., к.т.н., доц. каф. МБІС		
Розділ III	Адлер О.О., к.т.н., доц. каф. МБІС		

7.Дата видачі завдання 22 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	22.03.2024	30.03.2024	Виконано
2	Аналіз методів розв'язання задачі	01.04.2024	10.04.2024	Виконано
3	Постановка задачі розробки програми	11.04.2024	18.04.2024	Виконано
4	Дослідження інформаційних джерел	19.04.2024	30.04.2024	Виконано
5	Аналіз варіантів роботи програми	01.05.2024	10.05.2024	Виконано
6	Розробка алгоритму роботи	11.05.2024	18.05.2024	Виконано
7	Реалізація програми	19.05.2024	25.05.2024	Виконано
8	Тестування програми	26.05.2024	28.05.2024	Виконано
9	Оформлення матеріалів до захисту БДР	29.05.2024	07.06.2024	Виконано

Студент _____ Бондарчук А.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Адлер О.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дипломна робота присвячена дослідженню методів приховування інформації у контурах об'єктів растрових зображень. Основною метою дослідження є розробка та впровадження ефективних алгоритмів стеганографії, які дозволяють надійно вбудовувати секретні дані в зображення, зберігаючи при цьому високу якість та візуальну непомітність змін

У роботі розглянуто теоретичні основи стеганографії, зокрема особливості та переваги використання контурів об'єктів для приховування інформації. Проаналізовано існуючі методи та алгоритми, їхні переваги та недоліки, а також перспективи їх удосконалення.

Додаток забезпечує конфіденційність даних, які приховані у контурах об’єктів растрових зображень.

Отримані результати можуть бути використані для захисту інформації у різних галузях, зокрема в цифровій комунікації, зберіганні даних та інших сферах, де важлива конфіденційність та цілісність інформації.

Ключові слова: стеганографія, растрові зображення, контури, біт, конфіденційність, пікселі, JPEG, PNG.

ABSTRACT

The thesis is devoted to the study of methods for hiding information in the contours of raster image objects. The main goal of the research is to develop and implement effective steganography algorithms that allow to reliably embed secret data in images while maintaining high quality and visual invisibility of changes.

The paper discusses the theoretical foundations of steganography, including the features and advantages of using object contours to hide information. Existing methods and algorithms, their advantages and disadvantages, as well as prospects for their improvement are analyzed.

The application ensures the confidentiality of data that is hidden in the contours of raster images.

The obtained results can be used to protect information in various fields, including digital communication, data storage, and other areas where confidentiality and integrity of information are important.

Keywords: steganography, raster images, contours, bit, privacy, pixels, JPEG, PNG.

ЗМІСТ

ВСТУП

У наш час аспект інформаційної безпеки і захист конфіденційної інформації є ключовим фактором інтересів під час розробки. Одним із методів, що набирають популярності, є приховування інформації у контурах об'єктів растрових зображень. Це відкриває нові можливості для захисту даних у візуальному контенті, який використовується у різних сферах веб-індустрії. Проте, багато існуючих методів залишаються вразливими до атак через недостатню прихованість інформації.

Це дослідження спрямоване на розробку програми для приховування інформації у контурах об'єктів растрових зображень, що дозволить підвищити рівень безпеки даних користувачів. Актуальність цього дослідження підкреслюється зростаючою потребою у захисті інтелектуальної власності та загальною необхідністю у вдосконаленні методів кібербезпеки.

У рамках дослідження буде проведено аналіз існуючих методів стеганографії, виокремлені їх переваги та недоліки, зокрема, недостатня ефективність при спробах виявлення прихованих даних. На основі цього аналізу буде запропоновано новий метод приховування інформації у контурах об'єктів, що поєднує переваги існуючих методів та підвищує рівень безпеки даних.

Далі в дослідженні розглядатимуться технічні аспекти реалізації програми, включаючи вибір мови програмування, розробку архітектури програми та теоретичну розробку алгоритмів приховування інформації у контурах об'єктів растрових зображень.

Метою даного дослідження є створення програми для приховування інформації, яка не лише забезпечить високий рівень безпеки, а й дозволить користувачам зберігати конфіденційність даних у візуальному контенті завдяки застосуванню сучасних стеганографічних методів.

Предметом дослідження є розробка та реалізація програми приховування інформації у контурах об'єктів растрових зображень, спрямованої на захист цифрових даних у різних творчих та професійних проектах. Основна увага приділяється аспектам забезпечення безпеки даних користувачів шляхом

використання стеганографічних методів.

Об'єктом дослідження є процес зберігання та обміну даними з використанням програми приховування інформації у контурах об'єктів растрових зображень, з метою захисту конфіденційності, цілісності та доступності даних під час їх передачі через мережу Інтернет. Важливою частиною дослідження є інтеграція стеганографічних методів у процес обробки та зберігання даних, а також аналіз ефективності цих методів у контексті забезпечення безпеки даних користувачів..

1 ТЕОРЕТИЧНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття стеганографії та приховування інформації

Стеганографія - це метод передачі або зберігання інформації таким чином, щоб пряма передача або зберігання секретних даних залишалися конфіденційними. Стеганографія вже давно використовується як засіб передачі та зберігання даних. До основних основних засобів можна віднести ті, які стають помітними тільки при певних умовах. Наприклад, завдяки спеціальному шаблону, що відображає необхідну інформацію при освітленні, нагріванні, впливі з'єднань або в поєднанні з текстом. Існують також певні паролі, і на зображенні є мікродота, зменшена в сотні разів, оскільки звичайні слова набувають прихованого значення.

Цифрова стеганографія з'явилася нещодавно, але вважається новою галуззю науки порівняно з класичною стеганографією. Цифрова стеганографія - це конфіденційна передача даних, як правило, цифрові відбитки пальців використовуються для захисту прав. Наприклад, при продажу електронних матеріалів. Для організації захисту до кожної копії прикріплюється спеціальний знак або цифровий відбиток. Якщо зловмисник все ще може взяти і підробити цифровий відбиток, його буде неможливо розкрити [1].

Але поки цього не станеться, ви не зможете поширювати дані, захищені так званим тегом "контейнер". Водяні знаки стеганографа використовуються для захисту авторських прав. При використанні водяного знака передбачається, що кожна копія контейнера має однакову стеганографічну мітку. Класична стеганографія спрямована на приховану передачу інформації. Суть в тому, що він відіграє головну роль у захисті контейнера. Найголовніше-це сам факт передачі конфіденційної інформації. Тому що безпосередньо існування даних не повинно бути конфіденційним, а факт передачі повинен бути конфіденційним. Для цих цілей необхідно розробити стегосистему на основі певних принципів.

– потенційні шахраї знають нюанси побудови меганографічної системи. Однак, оскільки він не має ключових даних, можна отримати

інформацію про існування та вміст конфіденційних даних.

– у шахраїв немає ключів, тому виявлення наявності важливих повідомлень не впливає на конфіскацію і крадіжку тих же повідомлень в інших даних.

– шахраї не повинні мати ніяких технічних, програмних або інших переваг при визначенні вмісту вбудованих даних.

Залежно від типу система класифікується наступним чином. Стегосистема, яка має відомий ключ або відкритий ключ і має приватний ключ або приватний ключ. Стегосистема із закритим ключем працює за цим принципом. Для обміну секретними повідомленнями, вбудованими в контейнер, закритий ключ повинен бути відкритий заздалегідь всім учасникам обміну або відправлений по захищеному каналу.

Слід зазначити, що метод цифрової обробки зображень також викликає значні труднощі в забезпеченні працездатності ЦВЗ. Чим краще метод стиснення зображень, тим менше варіантів доступно для розміщення додаткової інформації. Розробка методу алгоритму стиснення зображень призвела до зміни уявлення про технологію реалізації ЦВЗ. Спочатку було запропоновано вбудувати інформацію в додатковий порожній біт, але інформація була поміщена в найбільш ефективну область зображення, і пошкодження цієї області біта призвело до спотворення самого зображення.

Отже, не випадково, що стегоалгоритми враховують властивості зорової системи людини (SLS) та властивості стиснення зображення. Алгоритм в основному використовує ті ж перетворення, що і сучасні алгоритми стиснення. Водночас, очевидно, існує 3 можливості. Ця інформація може бути вбудована в оригінальне зображення одночасно зі стисненням зображення контейнера або в попередньо стиснене зображення у форматі JPEG. Таким чином, враховуються особливості людського мозку та його аспекти в алгоритмах стиснення зображень. Виконання лінійного ортогонального перетворення зображення є трудомістким обчислювальним процесом, хоча існує швидке рішення. Таким чином, в деяких випадках структура входження в просторовій області

зображення може бути обмежена.

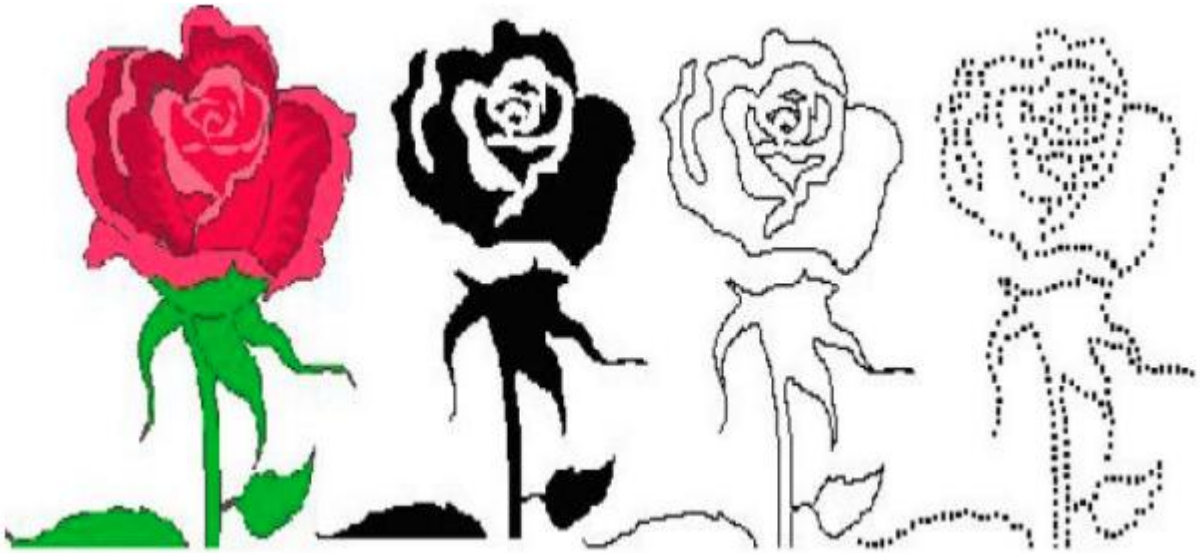


Рис. 1.1 - знаходження контуру

Для розуміння взаємодії стенографії з методами захисту зображень, необхідно розуміти як влаштоване зображення. Адже растрове зображення-це зображення, що представляє набір точок (пікселів), що використовуються для візуалізації на моніторі. Він має роздільну здатність, яка включає фізичну довжину монітора та кількість пікселів у його ширині. Інформаційна вага (Розмір, об'єм) растрового зображення дорівнює кількості пікселів та інформаційній вазі 1 пікселя. Отже, якщо n - ширина растрового зображення, M - довжина растрового зображення, а Q - вага інформації в 1 піксель, то V-вага інформації всього зображення дорівнює:

$$V = N * M * \mathrm{Q}$$

На зорі його появи растрове зображення було чорним і білий, що означає, що пікселі були виділені або залишилися чорними. Це означає, що для кодування інформації про стан пікселя потрібно лише 1 біт машинного слова, яке приймає значення одиниці 1 або нуль 0. Параметри, що характеризують кількість бітів, що використовуються для опису 1 пікселя, називаються глибиною кольору, якістю передачі або кількістю бітів зображення [2].

Наприклад, можуf записувати чорно-білі зображення з глибиною кольору k = 1, де n = 1000 пікселів, M = 500 пікселів, пам'ять V =1000 *500 * 1 = 500 С., 000. Ці технології давно зробили крок вперед, і зараз неможливо уявити таку

фігуру, тому що образ змінився.

Монохромне зображення - це не просто чорно-біле зображення з глибиною кольору k = 1. Значення K> 1. Оскільки еволюція монохромних зображень почалася з глибини кольору k = 1, і з'явилося зображення k = 2, і було 2 = 2 = 4, було 4 можливих відтінку. Тобто колір 1 пікселя може бути закодований за допомогою 2 біт, які можуть приймати значення 1 з 2 кодів в двійкових системах (1 або 0), і 2 біт, які можуть приймати значення 1 з 4 кодів в двійкових системах.(00, 01, 10, 11). Потім стало можливим створювати зображення з якістю передачі k = 3, K = 4, K = 8, K = 16.

Рис. 1.2 - пікселі як одиниця графіки

У поданні користувача для правильного відображення досить використовувати зображення з бітовою глибиною k = 8.Збільшення глибини кольору призвело до нереального збільшення кількості можливих кольорів.

З'явилися кольорові схеми, а червоні, зелені та сині кольорові компоненти

- модель RGB - стали використовуватися для представлення елементів палітри. Для кольорових зображень з глибиною k = 8 біт можна використовувати 256 різних кольорів. Червоному компоненту кольору присвоєно лише 3 біти, тобто 2 3 = 8, а червоному компоненту кольору присвоєно лише 3 біти, тобто 2 3 = 8A, а червоному компоненту кольору присвоєно лише 3 біти, тобто 2 3 = 8 комірок. використовується в зображенні [3].

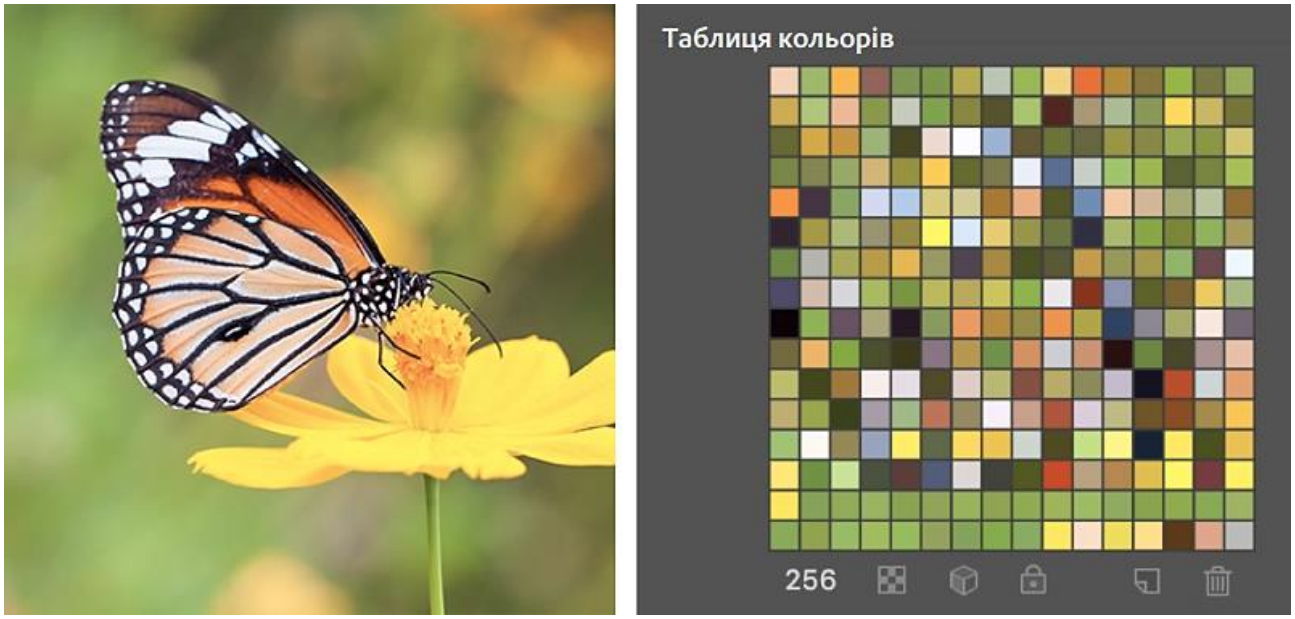


Рис. 1.3 - бітова глибина і її параметри

Синій компонент мав 2 біти або 4 можливі відтінки. Кольорове зображення з глибиною K = 12 біт може мати 2-4 різних кольори, оскільки кожному колірному компоненту, що відповідає компоненту 16 = 4 тони шкіри, присвоюється 4096 різних кольорів. Зображення з глибиною кольору 12 біт іноді використовуються в простих пристроях з кольоровими дисплеями. Використання глибини кольору k = 15, k = 16 і K = 18 біт в зображенні називається технологією високих кольорів. Найбільш придатними для сприйняття і ближчими до реальних кольорів є зображення з глибиною кольору k = 24, Що означає, що кожен колірний канал має довжину 8 біт і, отже, має 28 = 256 різних відтінків, які може представляти кожен канал. Метод нанесення цього кольору називається справжнім кольором і включає можливість нанесення 16777216 кольорів. Наприклад, растрове зображення з глибиною кольору k = 1000 пікселів має N = 500 пікселів, M = 24 пікселів, пам'ять V=1000 * 500 * 24 =

12000000 біт або 1,5 МБ. Існує багато форматів зображень, деякі підходять для пошуку оптимального співвідношення якості та розміру, інші підходять для відновлення зображень.

Рис. 1.4 – растрові зображення

Головною особливістю формату є можливість стиснення зображень. Стиснення може бути виконано 2 способами: низька якість і висока якість. Стиснене зображення без втрати якості зберігає стиснуту інформацію, тому оригінальне зображення можна відновити без стиснення. Неможливо відновити оригінальне зображення зі стисненого зображення без втрати якості, але воно набагато менше, ніж стиснене зображення без втрати якості. Ось список можливих форматів зображень:

– Формат BMP. Особливістю цього формату є те, що BMP був одним з перших графічних форматів.

– необроблений формат. Файли в цьому форматі містять вихідну

інформацію в матриці і не обробляються процесором камери. Цей формат надає великі можливості для обробки зображень, оскільки він зберігає максимальну якість і може бути стиснутий без втрати якості. Розширення таких файлів залежить від фотообладнання, на якому було знято зображення, оскільки різні виробники фотообладнання додали власне розширення формату RAW

– Формат JPEG. Найважливішою особливістю та визначальною характеристикою цього формату є його гнучкість, що означає широкий грудень стиснення зображень від максимального до максимального розміру.

– Формат JPEG2000. Цей формат дуже новий і має багато переваг у порівнянні з попереднім форматом jpeg. При тій же якості зображення Розмір файлу в форматі JPEG набагато більше.Основна відмінність між цими форматами полягає в методі стиснення Дека. Це метод штампування у форматі JPEG, при якому зображення розбивається на відмінні кадри з сильним стисненням, якого не відбувається при стисненні у форматі Jpeg2000. В даний час цей формат широко не використовується.

– Формат GIF. З появою Інтернету популярність цього формату значно зросла, і його стали використовувати для обміну зображеннями, оскільки Розмір файлу цього формату був більш компактним, ніж розмір інших форматів того часу. Підтримка анімації і стиснення без втрати якості. Недоліком є те, що він може зберігати зображення з дуже низькою глибиною кольору (8 біт).

– Формат PNG. Цей формат був створений для заміни формату gif. Він також стискається без втрат, але очевидною перевагою цього формату є те, що він може зберігати необмежену кількість кольорів і зберігати непрозорість.

– Формат TIFF. Цей формат в основному використовується для друку, оскільки він зберігає зображення в різних колірних просторах і має велику глибину кольору. Стиснення не погіршує якість, тому ці файли мають такий великий розмір.

– Формат PSD. При роботі в Photosh деці цей формат використовується для збереження проміжних результатів трудомісткої обробки зображень. Він має можливість зберігати зображення шарами, з різною глибиною кольору і в різних

колірних просторах. Великий обсяг інформації, що міститься в файлах цього формату, значно збільшується в розмірах, оскільки стиснення призводить до втрати якості [3].

1.2 Аналіз актуальності приховування інформації у контурах об’єктів растрових зображень

В умовах сучасного інформаційного суспільства питання захисту даних стає все більш важливим. Розвиток цифрових технологій та збільшення обсягів інформації, що передається через Інтернет, призводять до зростання потреби у надійних методах забезпечення конфіденційності. Один із перспективних напрямків у цій галузі – розробка програм для приховування інформації у контурах об’єктів растрових зображень.

Основні переваги методу

Приховування інформації в контурах об'єктів растрових зображень (стеганографія) є ефективним способом захисту даних, оскільки воно дозволяє інтегрувати секретну інформацію без значної зміни візуального вигляду зображення. Це ускладнює виявлення прихованих даних зловмисниками. Традиційні методи стеганографії часто вразливі до різних атак, але використання контурів об'єктів для вбудовування інформації забезпечує додатковий рівень прихованості та надійності.

У сучасному інформаційному суспільстві питання захисту даних набуває все більшої ваги. З розвитком цифрових технологій та збільшенням обсягів інформації, що передається через Інтернет, виникає нагальна потреба у надійних методах забезпечення конфіденційності. Один із перспективних напрямків у цій галузі – розробка програм для приховування інформації у контурах об’єктів растрових зображень. Цей метод має низку переваг, що роблять його особливо ефективним.

Це дозволяє інтегрувати секретну інформацію без значної зміни візуального вигляду зображення. Це значно ускладнює виявлення прихованих даних зловмисниками. Зокрема, традиційні методи стеганографії, що зазвичай

використовують зміни в кольорових каналах пікселів, можуть бути виявлені при ретельному аналізі. Однак використання контурів об’єктів для вбудовування інформації забезпечує додатковий рівень прихованості та надійності, оскільки контури є менш очевидними для аналізу та змін [4].

Цей метод також розширює можливості стеганографії, пропонуючи альтернативу традиційним методам, які можуть бути вразливими до атак. Вбудовування інформації в контури об'єктів дозволяє уникнути багатьох недоліків стандартних підходів і підвищити ефективність приховування інформації. Зловмисникам стає набагато важче виявити чи видалити приховані дані без значного спотворення самого зображення.

Крім того, розробка таких програм є актуальною у контексті захисту інтелектуальної власності. Для творців контенту, таких як художники, дизайнери та фотографи, важливо забезпечити захист авторських прав. Програми приховування інформації можуть використовуватися для вбудовування водяних знаків, що допомагає у визнанні авторства та захисті від незаконного копіювання.

Сучасні медіаплатформи активно використовують зображення для передачі інформації. Забезпечення конфіденційності та цілісності даних, вбудованих у ці зображення, є актуальним завданням.

Зростаючий обсяг візуального контенту є однією з ключових характеристик сучасного цифрового суспільства. Медіаплатформи, такі як соціальні мережі, новинні сайти та розважальні сервіси, активно використовують зображення для передачі інформації, оскільки візуальний контент є більш привабливим і легким для сприйняття. У такому контексті забезпечення конфіденційності та цілісності даних, вбудованих у ці зображення, стає надзвичайно актуальним завданням.

Це пов'язано з кількома важливими аспектами:

– Обсяг та поширеність візуального контенту: кількість зображень, що щоденно передається через Інтернет, продовжує зростати. Ці зображення можуть

містити як загальнодоступну, так і конфіденційну інформацію, що потребує захисту від несанкціонованого доступу та маніпуляцій.

– Уразливість до атак: зображення можуть бути легко піддані різним формам атак, включаючи маніпуляції, фальсифікації та несанкціоноване використання. Це ставить під загрозу як конфіденційність, так і цілісність даних, вбудованих у візуальний контент.

– Необхідність у прихованні даних: в умовах, коли зображення широко використовуються для комунікації, існує потреба у прихованні важливих даних таким чином, щоб вони залишалися непомітними для сторонніх осіб. Це особливо важливо для забезпечення приватності та захисту інтелектуальної власності.

Використання контурів об'єктів для приховування інформації має значні переваги над традиційними методами стеганографії, що засновані на змінах у кольорових каналах пікселів. Традиційні методи, такі як Least Significant Bit (LSB) модифікація, можуть бути відносно легко виявлені при ретельному аналізі, оскільки вони залишають певні сліди, які можуть бути помічені за допомогою статистичних методів або аналізу аномалій у кольорових каналах.

Натомість, використання контурів об'єктів для приховування інформації має переваги. Зміни, внесені в контури об'єктів, менш помітні для людського ока порівняно зі змінами у кольорових каналах пікселів. Це дозволяє зберігати візуальну цілісність зображення, не викликаючи підозр у глядача.

Метод приховування інформації у контурах менш вразливий до виявлення за допомогою стандартних методів аналізу зображень. Контури за своєю природою містять різкі зміни яскравості, що ускладнює виявлення прихованих даних, оскільки вони можуть бути замасковані природними особливостями зображення.

Контури об'єктів можуть мати значну кількість точок для приховування даних, що дозволяє вбудовувати більше інформації без суттєвого впливу на якість зображення. Це робить метод ефективним для передачі великих обсягів конфіденційних даних [6].

Метод приховування інформації у контурах може бути застосований до широкого спектру зображень, включаючи фотографії, графічні зображення та ілюстрації. Це універсальність робить його привабливим для використання у різних сферах, від захисту інтелектуальної власності до забезпечення конфіденційності комунікацій. Використання контурів як носіїв прихованої інформації ускладнює завдання для зловмисників, оскільки їм необхідно точно знати, де шукати приховану інформацію і як її виявити. Це додає додатковий рівень захисту, підвищуючи безпеку переданої інформації.

Таким чином, використання контурів об'єктів для стеганографії пропонує більш високий рівень прихованості та захисту даних порівняно з традиційними методами, роблячи його перспективним напрямком для подальшого розвитку у сфері інформаційної безпеки.

Розширення можливостей стеганографії є важливим напрямком у сучасних дослідженнях з інформаційної безпеки. Традиційні методи стеганографії, що використовують зміни у кольорових каналах пікселів, мають певні обмеження та вразливості. Оскільки такі методи вносять зміни безпосередньо в кольорову структуру зображення, вони можуть бути відносно легко виявлені при ретельному аналізі, особливо за допомогою сучасних інструментів для аналізу зображень.

Використання контурів об’єктів для стеганографії дозволяє уникнути цих недоліків і пропонує ряд важливих переваг:

Метод приховування інформації у контурах об’єктів дозволяє зберегти візуальний вигляд зображення практично незмінним. Контури об’єктів часто містять природні різкі зміни яскравості, що дозволяє вбудовувати інформацію без значного спотворення зображення.

Завдяки використанню контурів об’єктів, стеганографія стає менш вразливою до виявлення за допомогою стандартних методів аналізу зображень. Зловмисникам значно складніше виявити зміни, внесені в контури, оскільки вони часто замасковані під природні характеристики зображення.

Контури об’єктів можуть містити більше даних, ніж традиційні методи, що використовують кольорові канали пікселів. Це дозволяє вбудовувати більші обсяги інформації без втрати якості зображення, що є важливим для забезпечення ефективного приховування даних.

Метод стеганографії, що використовує контури об’єктів, може бути застосований до різних типів зображень, включаючи фотографії, графічні ілюстрації та навіть скановані документи. Це розширює можливості використання стеганографії у різних галузях, від цифрового мистецтва до забезпечення конфіденційності у бізнес-комунікаціях.

Використання контурів об’єктів робить стеганографію більш надійною, оскільки інформація може бути вбудована в такі частини зображення, які менше піддаються змінам при обробці або стисненні зображень. Це забезпечує кращу збереженість прихованої інформації у разі обробки зображення.

Розширення можливостей стеганографії через використання контурів об’єктів відкриває нові перспективи у сфері захисту даних. Цей підхід не лише підвищує ефективність приховування інформації, але й робить процес більш надійним і універсальним. У сучасному світі, де захист інформації стає все більш критичним, такі інноваційні методи можуть значно підвищити рівень безпеки цифрових даних.

Для творців контенту, таких як художники, дизайнери та фотографи, важливо забезпечити захист авторських прав. Програми приховування інформації можуть використовуватися для вбудовування водяних знаків, які допомагають у визнанні авторства та захисті від незаконного копіювання. У сучасному світі кіберзагроз, необхідність у нових методах захисту даних є надзвичайно важливою. Стеганографічні методи приховування інформації можуть слугувати додатковим інструментом у загальній стратегії забезпечення кібербезпеки [7].

1.3 Аналіз існуючих аналогів програм захисту інформації

Розглянемо список п'яти програм для приховування інформації у контурах об'єктів растрових зображень:

– Steganography Studio - це програмне забезпечення дозволяє вбудовувати повідомлення в растрові зображення, аудіо- та відеофайли.

– OpenStego - програма також пропонує можливість приховувати інформацію у растрових зображеннях, проте, вона є відкритою для використання.

– SilentEye - ще один відкритий інструмент для стеганографії, який може бути використаний для приховування інформації у растрових зображеннях.

– Hide in Picture - програма, як і інші, дозволяє вбудовувати текстову інформацію або файли в растрові зображення.

– Steghide: Steghide - інструмент командного рядка для стеганографії, який дозволяє вбудовувати інформацію в растрові та аудіофайли.

Steganography Studio - це програмне забезпечення, призначене для приховування повідомлень у растрових зображеннях, аудіо- та відеофайлах. Воно володіє простим інтерфейсом користувача, що спрощує процес приховування інформації, і має ряд корисних функцій для стеганографії [8].

Переваги:

– Простий у використанні інтерфейс, що полегшує вбудовування інформації у зображення.

– Можливість приховувати повідомлення в різних типах медіафайлів, включаючи зображення, аудіо та відео.

– Розширені функції для настройки процесу стеганографії, такі як налаштування рівня компресії чи вибір методу вбудовування.

Недоліки:

– Може бути обмежений у функціональності порівняно з іншими програмами стеганографії, які можуть мати більше додаткових функцій або опцій.

– Інформація, прихована за допомогою цієї програми, може бути виявлена деякими аналізаторами медіафайлів, що може становити ризик для конфіденційності.

Загалом, Steganography Studio - це корисний інструмент для проведення стеганографічних операцій з простим і зрозумілим інтерфейсом, хоча він може бути менш розширеним у порівнянні з деякими іншими програмами у цій категорії.

OpenStego - це відкритий інструмент для стеганографії, який дозволяє приховувати повідомлення у растрових зображеннях. Він має простий, але функціональний інтерфейс та набір інструментів для проведення стеганографічних операцій.

Переваги:

– Відкритий джерела, що означає, що ви можете переглядати його вихідний код та при необхідності адаптувати під власні потреби.

– Простий і зрозумілий інтерфейс користувача, який спрощує процес вбудовування інформації у зображення.

– Можливість приховувати різноманітну інформацію у растрових зображеннях без значного зниження їх якості.

Недоліки:

– Може не мати таких розширених функцій, як деякі комерційні або більш спеціалізовані інструменти.

– Можливість виявлення прихованої інформації деякими аналізаторами медіафайлів, особливо якщо використовується проста методика вбудовування.

Загалом, OpenStego є ефективним інструментом для проведення стеганографічних операцій з відкритим вихідним кодом та простим інтерфейсом, хоча він може бути менш розширеним у порівнянні з деякими іншими програмами у цій категорії.

SilentEye - це відкрите програмне забезпечення для стеганографії, яке дозволяє приховувати повідомлення у растрових зображеннях та інших

медіафайлах. Він володіє інтуїтивним і зручним інтерфейсом користувача, що робить його привабливим для початківців та експертів одночасно.

Переваги SilentEye:

– Простий і зрозумілий інтерфейс: Для багатьох користувачів SilentEye стає зручним використанням завдяки своєму простому інтерфейсу.

– Відкритий вихідний код: Будучи відкритим програмним забезпеченням, SilentEye забезпечує високий рівень прозорості і безпеки для користувачів.

– Розширена функціональність: Ця програма надає користувачам різні можливості для приховування повідомлень у різних типах медіафайлів.

– Універсальність: SilentEye може бути корисним для широкого спектру користувачів, від початківців до експертів.

– Безкоштовний та безкоштовний у використанні: SilentEye доступний для використання безкоштовно, що робить його доступним для всіх.

Недоліки SilentEye:

– Можливість виявлення прихованої інформації: Як і у багатьох програм стеганографії, інформація, прихована за допомогою SilentEye, може бути виявлена деякими аналізаторами медіафайлів.

Програма Hide in Picture має деякі цікаві особливості.

Hide in Picture відомий своєю простотою використання. Це дозволяє користувачам без особливих технічних навичок швидко і легко приховувати повідомлення у растрових зображеннях.

Крім текстової інформації, Hide in Picture також дозволяє приховувати файли різних типів, такі як документи, відео або аудіо, у растрових зображеннях. Це робить його більш універсальним і функціональним інструментом для стеганографії.

Програма володіє високою ефективністю роботи, що дозволяє швидко вбудовувати інформацію у зображення без зайвих затримок чи складних налаштувань. Програма зазвичай зберігає якість растрового зображення незмінною після приховування інформації, що робить приховані дані

малопомітними для звичайного спостерігача. Програма пропонує простий та ефективний спосіб стеганографії з можливістю приховувати різноманітну інформацію у растрових зображеннях зберігаючи їхню якість.

Steghide - це інструмент командного рядка для стеганографії, який відомий своєю простотою та ефективністю вбудовування інформації у різні типи медіафайлів. Особливості Steghide:

– Командний рядок. Steghide працює через командний рядок, що робить його особливо корисним для технічно обізнаних користувачів, які шукають більше контролю над процесом стеганографії.

– Різні типи медіафайлів. Він підтримує вбудовування інформації у різні типи медіафайлів, включаючи растрові зображення та аудіофайли, що робить його більш універсальним інструментом.

– Ефективність і швидкість. Steghide володіє високою ефективністю та швидкістю роботи, що дозволяє швидко і ефективно вбудовувати інформацію у медіафайли.

– Безпека. Використання командного рядка може забезпечувати додатковий рівень безпеки, оскільки користувачі можуть контролювати доступ до програми та її вхідного і вихідного потоків.

Отже, Steghide є потужним інструментом для стеганографії з використанням командного рядка, який володіє ефективністю та простотою використання, особливо для тих, хто знає, як користуватися командним рядком [9].

1.4 Висновки та постановка задач

У цьому розділі було проведено аналіз предметної області та різноманітних сфер застосування технологій приховування інформації у контурах об'єктів растрових зображень. Такий ретельний огляд дозволив краще розібратися з важливістю та широким спектром використання таких технологій у сучасному цифровому світі.

Обґрунтовано актуальність проблеми зберігання конфіденційної інформації у растрових зображеннях та її захисту. Розглянуті існуючі аналоги програм з приховування інформації, де виявлені як переваги, так і недоліки. Зокрема, підкреслено вразливість таких програм у зв'язку з відсутністю надійного шифрування, що створює загрозу розкриття конфіденційної інформації у разі несанкціонованого доступу.

Проведений аналіз дав змогу визначити переваги та недоліки існуючих аналогів, а також обрати оптимальні рішення для розробки власної програми з приховування інформації у растрових зображеннях. Підсумковий висновок полягає у розробці програмного забезпечення, яке міститиме переваги усіх інших програм та забезпечить надійний захист конфіденційної інформації у растрових зображеннях.

У процесі проведення аналізу теоретичного матеріалу, було визначено та постановлено задачі:

- розробка алгоритму роботи програми;
- здійснення програмної реалізації додатку;
- перевірка працездатності додатку.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

2.1 Вибір методу приховування для додатку

Вибір методу приховування для додатку є критичним етапом у розробці програмного забезпечення, яке працює з приховуванням інформації в зображеннях. Одним з таких методів є алгоритм на основі Least Significant Bit (LSB) та виявлення контурів. Розглянемо детально кожен з цих методів та обґрунтуємо їх вибір.

Алгоритм на основі LSB

Метод LSB (Least Significant Bit) приховування інформації в зображеннях є одним з найпоширеніших і простих методів стеганографії. Принцип методу полягає в заміні найменш значущих бітів пікселів зображення на біти приховуваної інформації. Цей метод має кілька переваг [10]:

– Простота реалізації: алгоритм LSB є простим для реалізації та не вимагає значних обчислювальних ресурсів.

– Мінімальні зміни зображення: зміни, внесені в зображення, не помітні для людського ока, оскільки модифікуються лише найменш значущі біти пікселів.

– Швидкість: процес приховування і вилучення інформації дуже швидкий, що робить цей метод ідеальним для додатків, які потребують високої продуктивності.

Переваги використання детектора контурів

– Приховування інформації в областях з високою текстурою або контурами робить її менш помітною для зловмисників.

– Використання контурів дозволяє ефективно використовувати доступний простір у зображенні для приховування інформації.

Алгоритм на основі дискретного косинусного перетворення (DCT)

Другим методом приховування інформації в зображеннях, є алгоритм на основі дискретного косинусного перетворення (DCT). Цей метод часто

використовується у форматах стиснення зображень, таких як JPEG, і має кілька важливих особливостей, які роблять його ефективним для стеганографії.

Принцип роботи методу DCT

Дискретне косинусне перетворення (DCT) перетворює зображення з просторової області в частотну область. Це дозволяє аналізувати і маніпулювати окремими частотними компонентами зображення. Метод DCT поділяє зображення на блоки (зазвичай 8x8 пікселів) і для кожного блоку виконує перетворення.

Після перетворення, значення компонентів можна змінювати для приховування інформації. Зазвичай, приховування здійснюється шляхом зміни значень високочастотних компонентів, що менш помітні для людського ока.

Переваги методу DCT

– Висока стійкість: зміна частотних компонентів робить приховану інформацію менш вразливою до виявлення та атак, таких як стиснення або фільтрація.

– Ефективність використання простору: метод DCT дозволяє приховувати більше інформації без значної втрати якості зображення.

– Застосовність у стиснених форматах: Цей метод ідеально підходить для використання у стиснених зображеннях, такцх як JPEG, оскільки він природно інтегрується в процес стиснення.

Процес приховування інформації на основі DCT складається з наступних етапів:

– Зображення розбивається на невеликі блоки розміром 8x8 пікселів.

– До кожного блоку застосовується дискретне косинусне перетворення, що переводить блок з просторової області в частотну.

– Вибрані високочастотні компоненти змінюються для приховування інформації.

– До кожного блоку застосовується обернене дискретне косинусне перетворення (IDCT), щоб повернути блоки у просторову область.

– Модифіковані блоки збираються разом, утворюючи кінцеве зображення з прихованою інформацією.

LSB підходить для простих та швидких реалізацій, де зміни у зображенні повинні бути мінімально помітними.

DCT є кращим вибором для випадків, де необхідна висока стійкість до атак та інтеграція у стиснені формати зображень, такі як JPEG.

Маючи два мотоди для роботи з різними типами зображень почнемо опис з алгоритму на основі LSB.

Спочатку розробимо функцію отримання пікселів зображення. Функція getImagePixels перетворює зображення на масив пікселів [11].

```
function getImagePixels($img) {
   $width = imagesx($img);
   $height = imagesy($img);
   $pixels = [];

   for ($y = 0; $y < $height; $y++) {
      for ($x = 0; $x < $width; $x++) {
         $pixels[$y][$x] = imagecolorat($img, $x, $y);
      }
   }

   return $pixels;
}
```

Тепер ми можемо перейти до виявлення контурів. Функція detectEdges використовує простий детектор контурів (наприклад, оператор Собеля) для визначення контурів у зображенні.

```
function detectEdges($pixels) {
   $height = count($pixels);
   $width = count($pixels[0]);
   $edges = [];

   for ($y = 1; $y < $height - 1; $y++) {
      for ($x = 1; $x < $width - 1; $x++) {
         $gx = ($pixels[$y - 1][$x - 1] & 0xFF) + 2 * ($pixels[$y - 1][$x] & 0xFF) + ($pixels[$y - 1][$x + 1] & 0xFF) -
($pixels[$y + 1][$x - 1] & 0xFF) - 2 * ($pixels[$y + 1][$x] & 0xFF) - ($pixels[$y + 1][$x + 1] & 0xFF);
         $gy = ($pixels[$y - 1][$x - 1] & 0xFF) + 2 * ($pixels[$y][$x - 1] & 0xFF) + ($pixels[$y + 1][$x - 1] & 0xFF) -
($pixels[$y - 1][$x + 1] & 0xFF) - 2 * ($pixels[$y][$x + 1] & 0xFF) - ($pixels[$y + 1][$x + 1] & 0xFF);

         $gradient = sqrt($gx * $gx + $gy * $gy);

         if ($gradient > 255) {
            $gradient = 255;
         }

         $edges[$y][$x] = $gradient;
      }
```

```
    }

    return $edges;
}
```

Далі можна розробити функціонал приховування повідомлення. Функція hideMessageInEdges перетворює повідомлення в двійковий формат і приховує його у найменш значущому біті пікселів, що складають контури об'єктів у зображенні.

```
function hideMessageInEdges($img, $edges, $message) {
    $width = imagesx($img);
    $height = imagesy($img);
    $binaryMessage = '';

    for ($i = 0; $i < strlen($message); $i++) {
        $binaryMessage .= str_pad(decbin(ord($message[$i])), 8, '0', STR_PAD_LEFT);
    }

    $messageLength = strlen($binaryMessage);
    $messageIndex = 0;

    for ($y = 0; $y < $height; $y++) {
        for ($x = 0; $x < $width; $x++) {
            if ($messageIndex >= $messageLength) {
                return;
            }

            if ($edges[$y][$x] > 128) {
                $color = imagecolorat($img, $x, $y);
                $r = ($color >> 16) & 0xFF;
                $g = ($color >> 8) & 0xFF;
                $b = $color & 0xFF;

                $b = ($b & 0xFE) | $binaryMessage[$messageIndex];
                $newColor = imagecolorallocate($img, $r, $g, $b);
                imagesetpixel($img, $x, $y, $newColor);

                $messageIndex++;
            }
        }
    }
}
```

Але існують різні способи більш надійного приховування інформації у зображеннях, ніж через зміну найменш значущого біту (LSB). Один із таких методів - використання дискретного косинусного перетворення (DCT) в частотній області зображення, як це робиться в JPEG-стеганографії.

Основні кроки методу:

– Перетворення зображення в частотну область за допомогою DCT.

– Вбудовування повідомлення в середні частоти (не в найвищі, щоб уникнути помітних змін, і не в найнижчі, щоб уникнути легкого видалення при стисненні).

– Зворотне перетворення DCT до просторової області.

Перетворення в градації сірого: Ми конвертуємо зображення в градації сірого для простоти обробки.

```
function rgb2gray($r, $g, $b) {
   return (0.299 * $r + 0.587 * $g + 0.114 * $b);
}
```

DCT та IDCT: Реалізація двовимірного дискретного косинусного перетворення (DCT) та його зворотного перетворення (IDCT).

```
function dct2($block) {
   $N = count($block);
   $dct = [];
   for ($u = 0; $u < $N; $u++) {
      $dct[$u] = [];
      for ($v = 0; $v < $N; $v++) {
         $sum = 0;
         for ($x = 0; $x < $N; $x++) {
            for ($y = 0; $y < $N; $y++) {
               $sum += $block[$x][$y] *
                  cos((2 * $x + 1) * $u * M_PI / (2 * $N)) *
                  cos((2 * $y + 1) * $v * M_PI / (2 * $N));
            }
         }
         $alpha_u = $u == 0 ? 1 / sqrt(2) : 1;
         $alpha_v = $v == 0 ? 1 / sqrt(2) : 1;
         $dct[$u][$v] = 0.25 * $alpha_u * $alpha_v * $sum;
      }
   }
   return $dct;
}


function idct2($dct) {
   $N = count($dct);
   $block = [];
   for ($x = 0; $x < $N; $x++) {
      $block[$x] = [];
      for ($y = 0; $y < $N; $y++) {
         $sum = 0;
         for ($u = 0; $u < $N; $u++) {
            for ($v = 0; $v < $N; $v++) {
               $alpha_u = $u == 0 ? 1 / sqrt(2) : 1;
               $alpha_v = $v == 0 ? 1 / sqrt(2) : 1;
               $sum += $alpha_u * $alpha_v * $dct[$u][$v] *
                  cos((2 * $x + 1) * $u * M_PI / (2 * $N)) *
                  cos((2 * $y + 1) * $v * M_PI / (2 * $N));
            }
         }
         $block[$x][$y] = 0.25 * $sum;
```

```
    }
  }
  return $block;
}
```

Приховування повідомлення: Ми приховуємо повідомлення у середніх частотах DCT-блоків, змінюючи значення відповідних коефіцієнтів.

```
function embedMessage($block, $message, $bitPos) {
  $binaryMessage = '';
  for ($i = 0; $i < strlen($message); $i++) {
    $binaryMessage .= str_pad(decbin(ord($message[$i])), 8, '0', STR_PAD_LEFT);
  }

  $messageLength = strlen($binaryMessage);
  $index = 0;

  for ($i = 1; $i < count($block) - 1; $i++) {
    for ($j = 1; $j < count($block[$i]) - 1; $j++) {
      if ($index >= $messageLength) {
        return $block;
      }
      $value = intval($block[$i][$j]);
      $bit = $binaryMessage[$index++];
      $block[$i][$j] = $value & ~(1 << $bitPos) | (intval($bit) << $bitPos);
    }
  }

  return $block;
}
```

Цей метод є більш стійким до різних видів атак та зменшення якості зображення, наприклад, при стисненні JPEG.

2.2 Розробка алгоритму додатка

Для початку розробки алгоритму необхідно описати архітектуру його роботи. Це виконується за допомогою створення блок-схеми адже це потужний інструмент для візуалізації, опису та аналізу різноманітних алгоритмів програм.

Блок-схеми дозволяють зобразити складні процеси або алгоритми у вигляді простих графічних елементів, що робить їх більш зрозумілими та доступними для аналізу.

Вони допомагають візуалізувати послідовність дій або рішень у процесі виконання завдання, що дозволяє аналізувати їх ефективність та правильність.

Блок-схеми дозволяють комунікувати складні ідеї та концепції шляхом графічного представлення, що сприяє співпраці між різними учасниками

проекту. Шляхом аналізу блок-схем можна виявити можливі проблеми, недоліки чи можливості для оптимізації процесів та алгоритмів.

Блок-схеми є важливим інструментом для навчання програмування, алгоритміки та логічного мислення, оскільки вони дозволяють студентам візуалізувати та розуміти складні концепції.

Вони мають багато корисних застосувань у різних галузях, починаючи від розробки програмного забезпечення до управління проектами та навчання [12].

За допомогою блок-схем можна зробити наступне:

– Візуалізувати складні процеси та алгоритми.

– Аналізувати послідовність дій у різних сценаріях.

– Виявляти потенційні проблеми та недоліки в процесах.

– Оптимізувати або вдосконалювати існуючі процеси.

– Комунікувати та співпрацювати з іншими учасниками проекту.

– Навчати та викладати основи програмування, алгоритміки та логічного мислення.

– Створювати документацію для проектів або програмного забезпечення.

– Документувати та пояснювати логіку програм або алгоритмів.

– Планувати та організовувати робочі процеси та проекти.

– Аналізувати та вдосконалювати ланцюжки постачання або виробничі процеси.

Для їх візуалізації використовується:

– прямокутник (або квадрат): представляє дії або операції, що виконуються, такі як введення/виведення даних, обробка інформації, операції над даними тощо.

– ромб: використовується для позначення умов або рішень у процесі, наприклад, перевірка умови (if-else).

– коло або овал: може використовуватися для позначення початку або кінця процесу, або для інших підпроцесів.

– лінія зі стрілкою: показує напрямок потоку програми або процесу.

– паралелограм: іноді використовується для позначення введення або виведення даних.

– трикутник: може використовуватися для позначення точок входу або виходу з підпрограм.

– трапеція: використовується для позначення обробки даних або дій, які відбуваються під певним умовом.

– ромб з двома входами або виходами: це може бути розширений ромб або два окремі ромби, які показують дві різні умови або результати.

– коробка з текстом: це просто контейнер для додаткового пояснювального тексту або коментарів.

– вертикальні чи горизонтальні лінії без стрілок: вони можуть використовуватися для показу зв'язків між різними частинами блок-схеми або позначення потоків даних.

– криві лінії: іноді вони використовуються для показу складних або нелінійних потоків в програмі або процесі.

– вигнуті стрілки або лінії: вони можуть використовуватися для показу відхилення від стандартного потоку програми.

– еліпс або круг з текстом всередині: це може використовуватися для позначення даних або змінних у процесі.

– лінії з переривчастими або точковими стрілками: вони можуть використовуватися для позначення зв'язків або взаємодії з іншими системами або компонентами.

– складена лінія або стрілка: використовується для показу складних або довгих зв'язків між різними елементами блок-схеми.

–

– коробка з символом "+" або "-": це може використовуватися для позначення об'єднання або розділення потоків даних або процесів.

– піраміда: може використовуватися для позначення ієрархії або структури даних.

– вигнуті прямокутники або ромби: вони можуть використовуватися для показу незвичайних умов або обробки даних, що відрізняються від стандартних умов.

– таблиця або сітка: може використовуватися для відображення складних структур даних або процесів з більшою кількістю деталей.

– коробка з підписом "loop": це може використовуватися для позначення циклічних або повторюваних дій у процесі.

– коробка з підписом "module" або "function": це може використовуватися для позначення підпрограм або функцій, які використовуються в процесі.

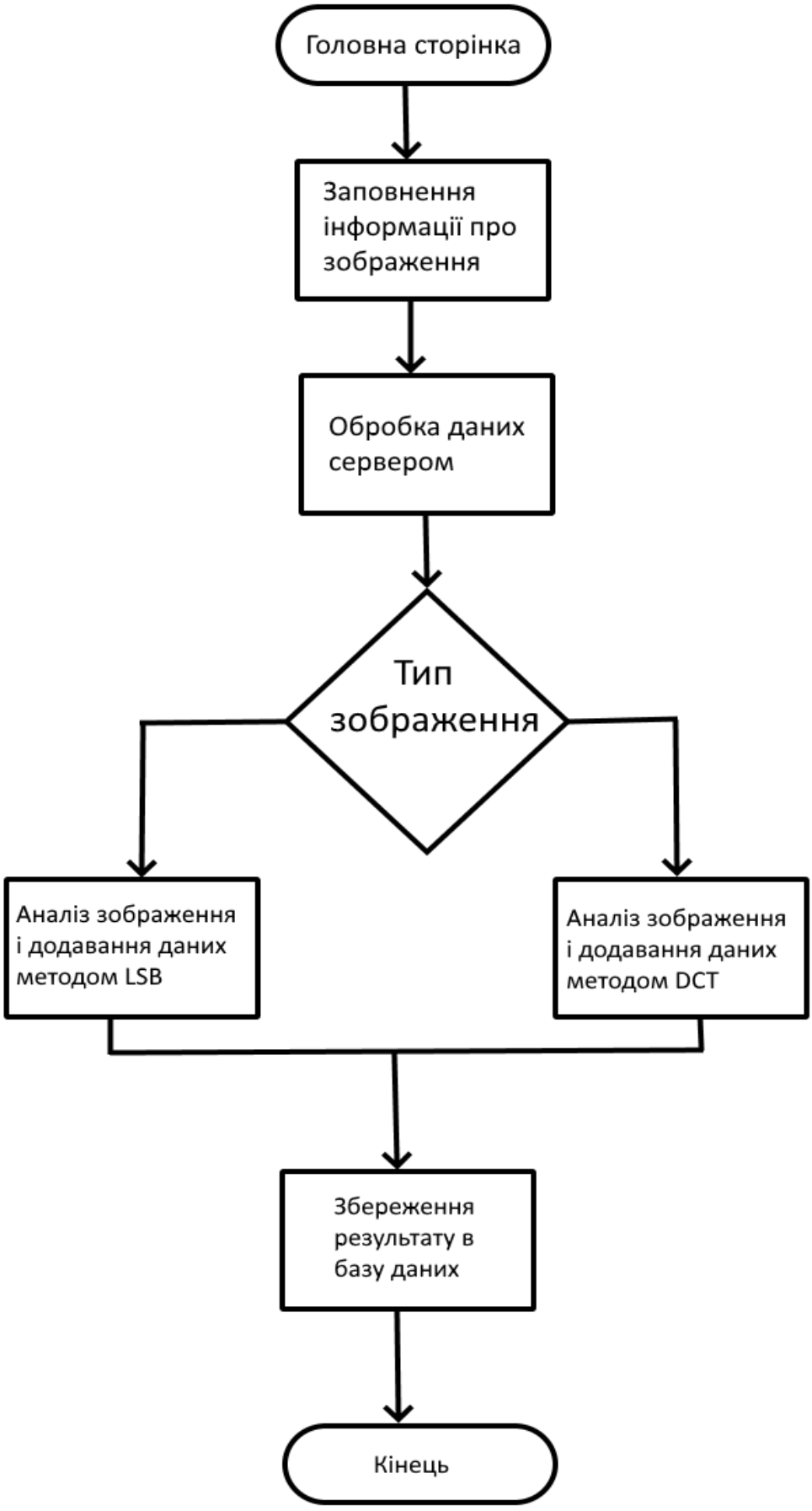


Рис. 2.1 - алгоритм роботи програми

Для проектування також використано шаблон MBC (Model-View-Controller) - інструмент, який ми використовуємо для забезпечення чіткого розділення відповідальностей у програмному забезпеченні. Його використання допомагає створювати більш структуровані, гнучкі та легко підтримувані додатки. Цей шаблон дозволяє чітко виділити компоненти, які відповідають за обробку даних (Модель), відображення інтерфейсу користувача (Вид) та обробку взаємодії користувача з додатком (Контролер). Такий підхід спрощує розробку, підтримку та масштабування нашого програмного забезпечення, а також робить код більш зрозумілим та легким для співпраці та обслуговування всієї команди розробників.

Шаблон проектування MBC є потужним інструментом в розробці програмного забезпечення, особливо для веб-додатків. MBC дозволяє чітко розділити логіку додатку на три основні компоненти: Модель, Вид та Контролер. Це сприяє збереженню чистоти коду та спрощує його розуміння та редагування. Модель і Контролер можуть бути повторно використані для різних інтерфейсів користувача. Наприклад, можна мати одну модель для додатку веб-сайту та іншу для мобільного додатку, при цьому залишаючи той самий контролер.

Кожен компонент MBC може бути тестований окремо. Модель може бути протестована для забезпечення правильності обробки даних, Вид для переконання в правильному відображенні інтерфейсу, а Контролер для переконання в правильному взаємодії між моделлю та видом.

Завдяки розділенню логіки на три компоненти, зміни в одному компоненті можуть відбуватися без впливу на інші. Це дозволяє зручно розширювати функціональність додатку або вносити зміни без ризику порушення роботи інших частин системи.

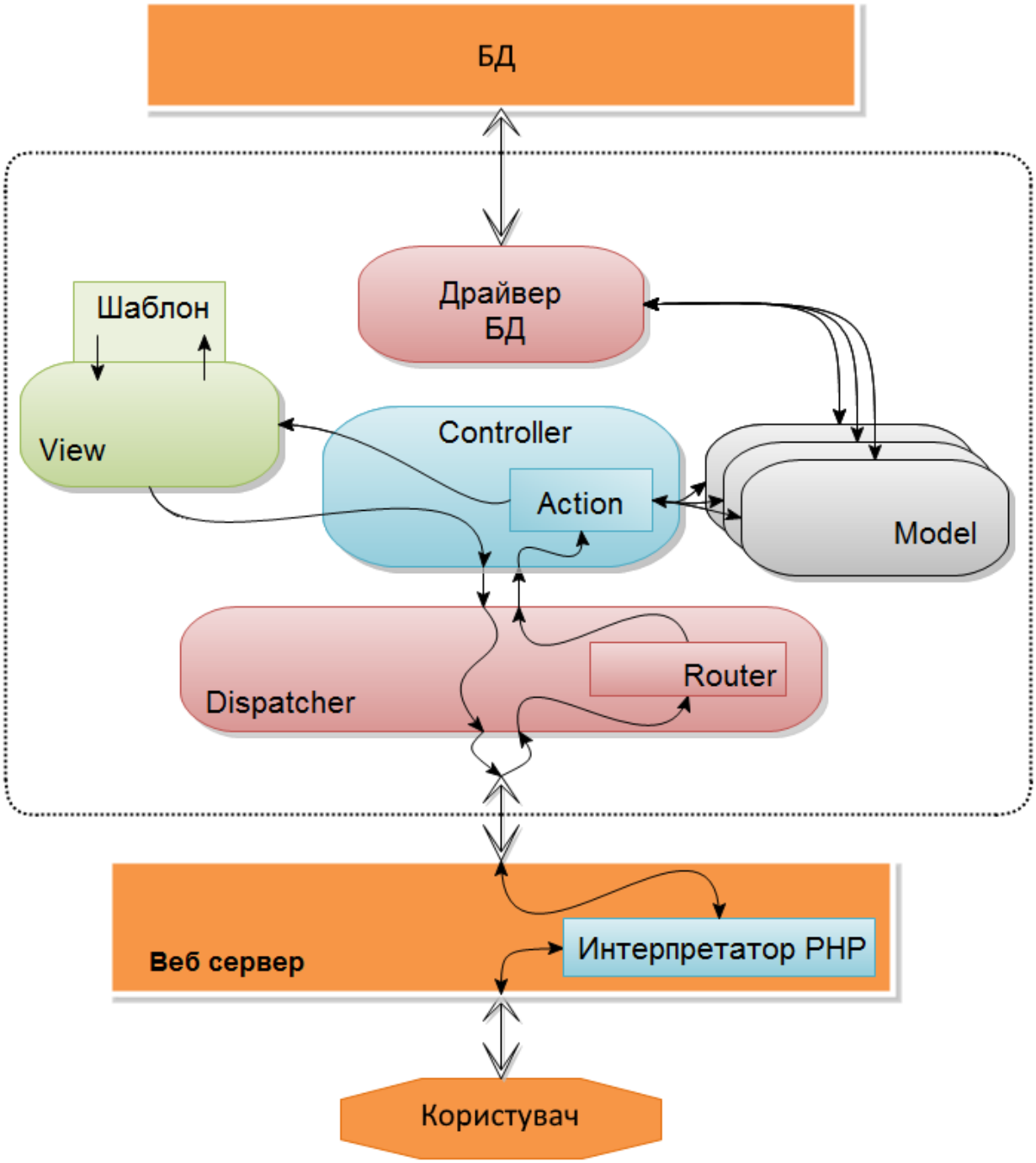


Рис. 2.2 - структура шаблону проектування

MBC є популярним шаблоном проектування, який багато розробників знають та розуміють. Використання стандартних підходів спрощує комунікацію між розробниками та підтримку коду в майбутньому. MBC є інструментом, який спрощує розробку, тестування та підтримку програмного забезпечення, зокрема веб-додатків [13].

2.3 Проектування алгоритму інтерфейсу

Перш ніж користувач зможе використовувати програму для приховування інформації в контурах зображень, він повинен авторизуватися в своєму акаунті. Це необхідно для збереження налаштувань, що стосуються приховування інформації у зображеннях.

Після успішного входу в акаунт користувач має можливість вибрати опцію завантаження растрового зображення для обробки. Вибравши зображення, користувач може завантажити файл для приховування інформації. Це виконується шляхом вибору файлу на комп'ютері та його завантаження на сервер. Також він має надати інформацію про авторство.

Після завантаження текстури користувач може переглянути її на вкладці "Зображення.

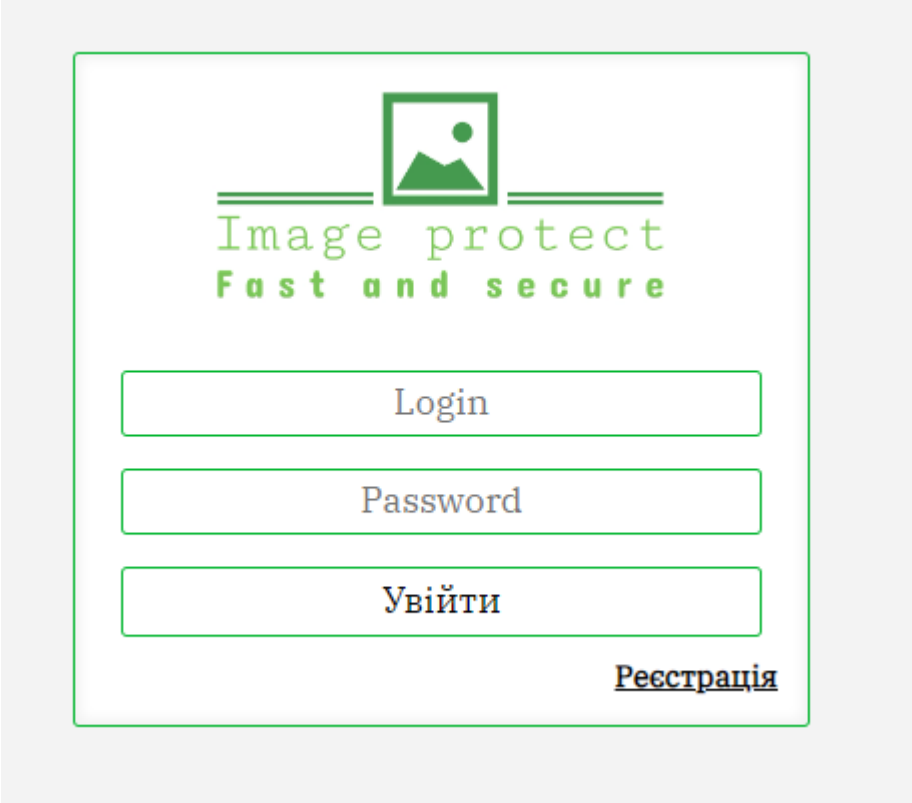


Рисунок 2.3 – авторизація користувача

Для нового користувача пропонується форма Реєстрації.

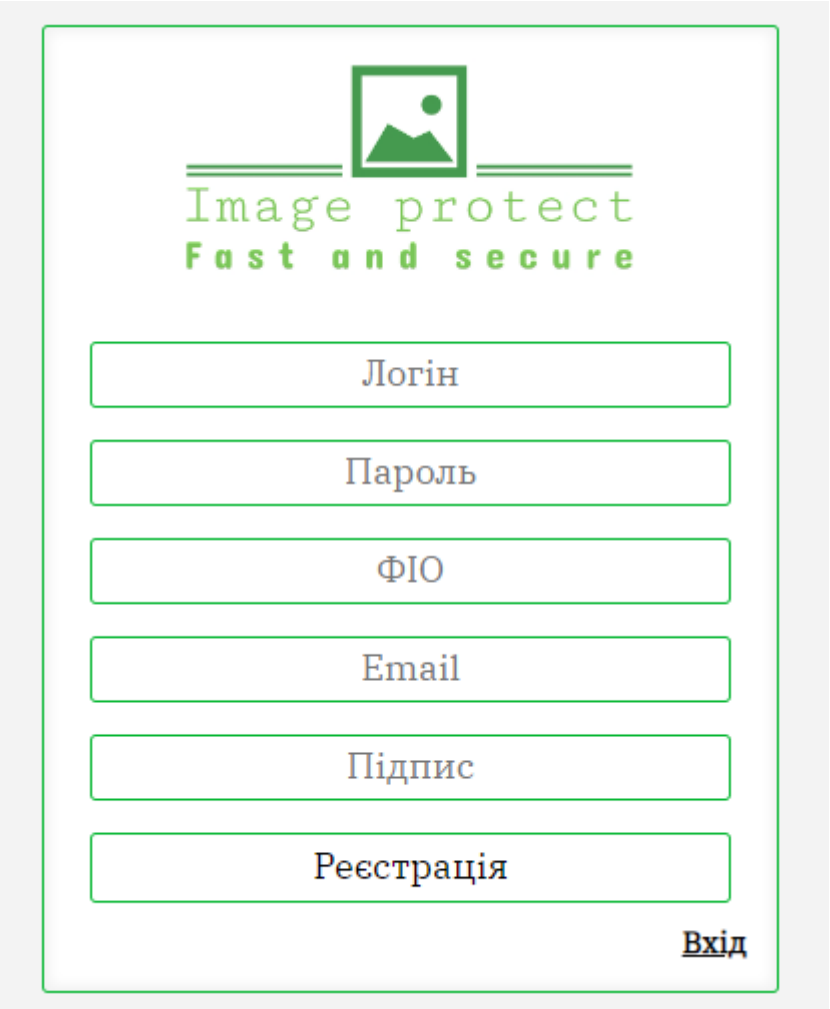


Рисунок 2.4 – реєстрація користувача

На сторінці «Зображення» відображаються всі завантажені файли. Можна переглянути назву файлу і дату вказану під час створення.

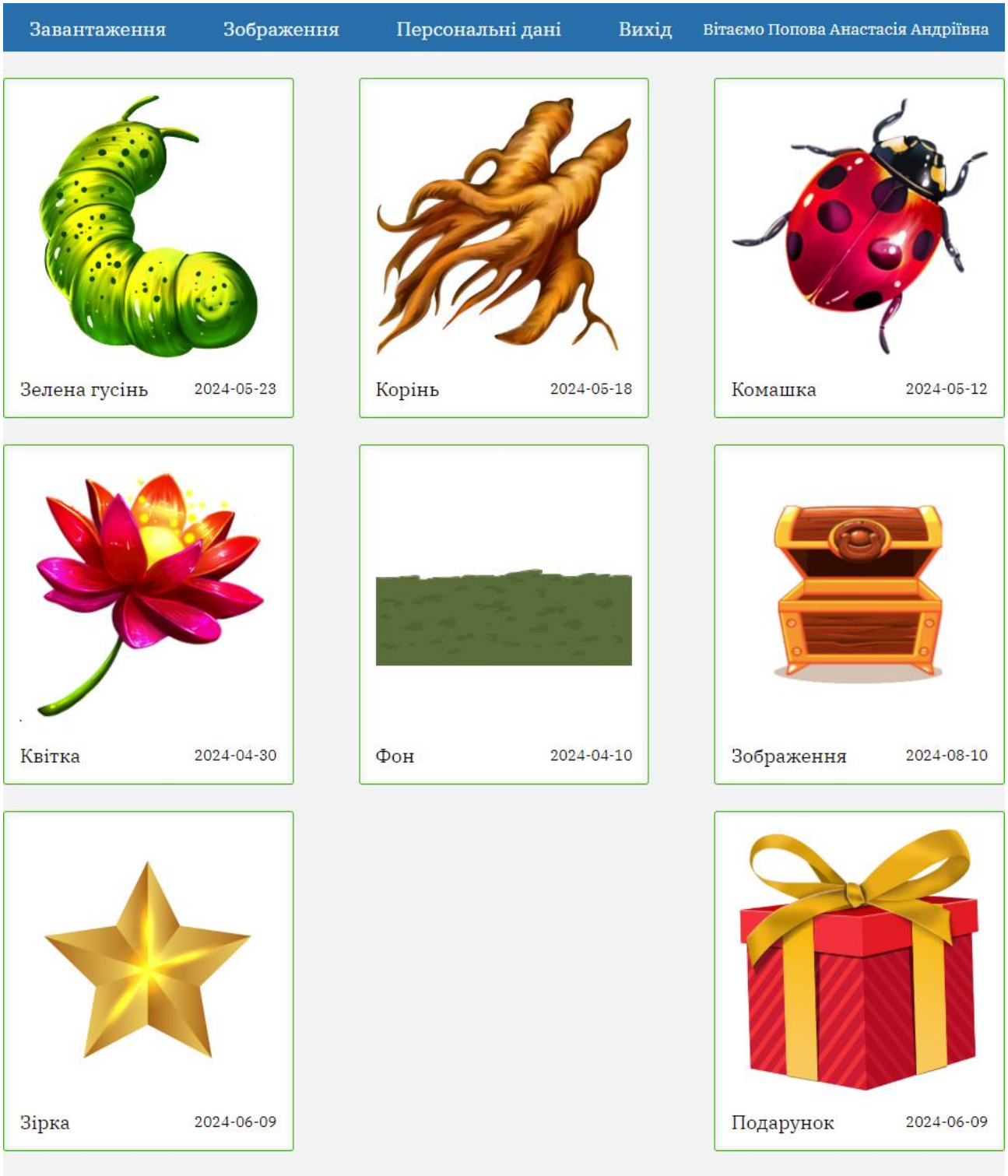


Рисунок 2.4 – файли користувача

Для завантаження нового зображення використовується форма веб-сторінки. У формі знаходяться інпути які приймають перелік параметрів для створення запису і додавання інформації до зображення.

```
<div class="container">
        <div class="menu">
                <div>
                        <a href="/user">Завантаження</a>
```

```
                    <a href="/user/info">Зображення</a>
                    <a href="/user/data">Персональні дані</a>
                    <a href="/user/logout">Вихід</a>
                </div>
                <p class="name">Вітаємо <?=$_SESSION['name']?></p>
            </div>
            <div class="take">
                <div class="right">
                    <p>PNG</p>
                    <span class="small">
                        Для PNG та інших форматів без втрат, LSB є гарним вибором завдяки простоті і мінімальному впливу на якість зображення.
                    </span>

                    <form class="c-form" action="/user/file" method="POST" enctype="multipart/form-data">
                        <input name="type" type="text" value="1" class="hide">

                        <span class="label">Дата</span>
                        <input name="date" type="date" >

                        <span class="label">Назва</span>
                        <input name="name" type="text" >

                        <span class="label">Текст</span>
                        <input name="text" type="text" >

                        <span class="label">Вартість</span>
                        <input name="sum" type="number" max="10000" min="1">

                        <span class="label">Завантажити зображення</span>
                        <input type="hidden" name="MAX_FILE_SIZE" value="512000000" />
                        <input name="load" type="file" required>

                        <input class="input-btn" type="submit" value="Приховати дані">
                    </form>
                </div>

                <div class="left">
                    <p>JPG</p>
                    <span class="small">
                        Для JPEG та форматів з втратами, DCT є кращим вибором через його стійкість до стиснення і здатність приховувати більше даних.
                    </span>

                    <form class="c-form" method="POST" action="/user/file" enctype="multipart/form-data">
                        <input name="type" type="text" value="2" class="hide">

                        <span class="label">Дата</span>
                        <input name="date" type="date" >

                        <span class="label">Назва</span>
                        <input name="name" type="text" >

                        <span class="label">Текст</span>
                        <input name="text" type="text" >

                        <span class="label">Вартість</span>
                        <input name="sum" type="number" max="10000" min="1">
```

```
                        <span class="label">Завантажити зображення</span>
                        <input type="hidden" name="MAX_FILE_SIZE" value="512000000" />
                        <input name="load" type="file" required>

                        <input class="input-btn" type="submit" value="Приховати дані">
                    </form>
                </div>
            </div>
        </div>
```

Рисунок 2.5 – форма для роботи PNG форматом

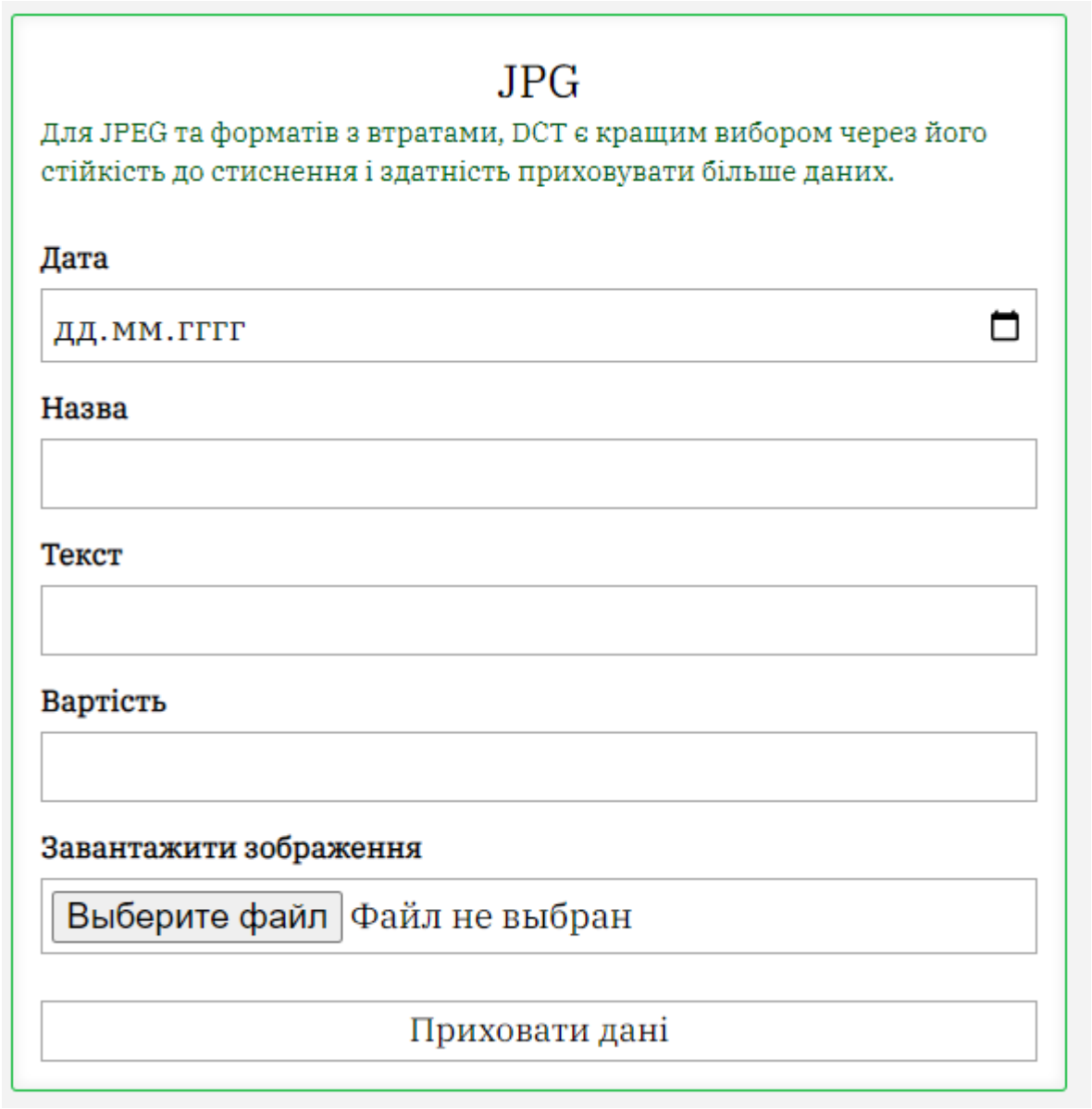


Рисунок 2.6 – форма для роботи JPG форматом

Після завантаження зображення дані передаються на сервер. Відбувається аналіз зображення і вставка інформації відповідним методом.

2.4 Висновки

Для створення програми приховування інформації у контурах об'єктів растрових зображень було обрано мову програмування PHP. Цей вибір зумовлений його гнучкістю та широким спектром інструментів для обробки зображень.

Архітектура програми базується на ідеї використання методу стеганографії, де інформація приховується у невидимих змінах пікселів

зображення. Клієнтська частина реалізована з використанням бібліотек для роботи з зображеннями та алгоритмів стеганографії для вбудовування та вилучення інформації. Серверна частина, також написана на PHP, відповідає за обробку запитів користувачів та забезпечення безпеки даних.

Для розробки програми приховування інформації у контурах об'єктів растрових зображень розглядали два основних варіанти: використання методу Дискретного Косинусного перетворення (DCT) та методу Найменш значущого біту (LSB). Обидва варіанти мають свої переваги та недоліки.

Архітектура програми буде побудована за принципом моделі-вид-контролер (MBC). Клієнтська частина програми буде відповідати за відображення та взаємодію з користувачем, використовуючи HTML, CSS та JavaScript у браузері. Серверна частина, написана на мові програмування PHP, забезпечуватиме обробку запитів користувачів, зберігання та обробку даних.

Використання DCT передбачає перетворення зображення у частотний домен, де інформація може бути прихована у менш важливих частотах. Цей метод забезпечує високу стійкість до атак на стеганографічний алгоритм, але може вимагати більшої обчислювальної потужності.

З іншого боку, використання LSB полягає в заміні найменш значущих бітів пікселів зображення на біти інформації. Цей метод простий у реалізації та ефективний для приховування менших обсягів даних, але може бути вразливим до стеганалізу.

Обираючи між DCT та LSB, враховуватимемо конкретні потреби проекту, рівень безпеки, а також обчислювальні вимоги. Враховуючи ці фактори, ми зможемо ефективно забезпечити приховування інформації у контурах об'єктів растрових зображень.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Вибір середовища розробки

PHP - це мова програмування, яка приваблює своєю простотою вивчення та використанням. Вона має досить простий синтаксис, що робить її дружньою для новачків і дозволяє швидко розпочати роботу над проєктами. Ще однією перевагою PHP є її широке застосування у веб-розробці. Вона використовується для створення веб-сайтів та веб-додатків і є основною мовою для таких популярних систем управління контентом, як WordPress, Drupal і Joomla. І нарешті, PHP має велику спільноту користувачів, що надає постійну підтримку, розвиток та оновлення мови. Це робить її привабливим вибором для багатьох веб-розробників у всьому світі. PHP-код зазвичай між тегами <?php та ?>. Вони використовуються для вказівки початку і кінця PHP-коду в HTML-документі [14].

Змінні в PHP оголошуються з символом $, наприклад $name або $age. В PHP є кілька основних типів даних, таких як рядки (string), цілі числа (integer), дійсні числа (float), логічні значення (boolean) та масиви (array).

PHP підтримує різні типи операторів, такі як арифметичні, порівняння, логічні та змінні. Оператори дозволяють виконувати різні дії над даними, такі як додавання, віднімання, порівняння та інші.

Умовні конструкції, такі як if, else, elseif і switch, дозволяють виконувати певний блок коду залежно від умови. Це дозволяє програмі реагувати на різні ситуації та приймати рішення на основі умов.

Цикли for, while і do-while дозволяють повторювати блоки коду деяку кількість разів або до виконання певної умови.

Масиви в PHP можуть бути індексованими або асоціативними. Вони дозволяють зберігати групу значень у зручній формі і легко керувати ними.

Функції в PHP дозволяють групувати код для виконання певних завдань. Вони можуть приймати аргументи, виконувати певні дії та повертати значення.

PHP підтримує об'єктно-орієнтований підхід, що дозволяє створювати класи і об'єкти з методами та властивостями [15].

Оператори в PHP дозволяють виконувати різні дії над даними, такі як арифметичні операції, порівняння, логічні операції тощо. Вони роблять код більш читабельним і ефективним, дозволяючи програмістам виражати складні логічні операції у короткій формі. Наприклад, оператори порівняння дозволяють порівнювати значення і виконувати певні дії в залежності від результату порівняння. Оператори також можуть бути використані для перевірки умов, виконання циклів, призначення значень змінним тощо.

Об'єктно-орієнтоване програмування (ООП) в PHP - це парадигма програмування, в якій програма структурується навколо об'єктів, які представляють реальні або абстрактні об'єкти з різними властивостями та методами. Ключові концепції ООП в PHP:

– Класи і об'єкти. Класи в PHP використовуються для визначення структури об'єктів. Вони містять властивості (змінні) та методи (функції), які описують поведінку об'єкта. Об'єкти є конкретними екземплярами класів.

– Інкапсуляція. Це концепція, що дозволяє обмежувати доступ до властивостей і методів об'єкта. В PHP це зазвичай досягається за допомогою модифікаторів доступу, таких як public, protected та private.

– Наслідування. Наслідування дозволяє класам успадковувати властивості та методи інших класів. Клас, що успадковується, називається дочірнім класом, а клас, від якого успадковується, - батьківським класом.

– Поліморфізм. Це можливість об'єктів одного класу використовувати методи, що визначені в інших класах. В PHP це зазвичай досягається за допомогою перевизначення методів у дочірніх класах.

– Абстракція. Абстракція в ООП використовується для приховання деталей реалізації і надання об'єктам лише необхідного інтерфейсу для взаємодії. В PHP це може включати використання абстрактних класів та інтерфейсів.

– Магічні методи. PHP надає спеціальні методи, які автоматично викликаються при певних подіях, таких як створення об'єкта (__construct),

знищення об'єкта (__destruct), виклик методу, який не був знайдений (__call), тощо.

Використання об'єктно-орієнтованого підходу у PHP дозволяє розробникам створювати більш структуровані, гнучкі та легко розширювані програми. ООП сприяє підтримці чистого коду, рефакторингу, тестуванню та іншим аспектам розробки програмного забезпечення [16].

Модульність дозволяє розділити програму на невеликі, автономні модулі (класи), які можна розглядати як окремі "будівельні блоки". Це полегшує розробку, тестування та підтримку коду, оскільки зміни в одному класі не впливають на інші частини програми. Спадкування це механізм успадкування дозволяє створювати нові класи на основі вже існуючих (батьківських) класів, успадковуючи їх властивості та методи. Це спрощує розробку, оскільки дозволяє використовувати та розширювати функціональність вже наявних класів. Поліморфізм в ООП дозволяє об'єктам різних класів вести себе по-різному у відповідь на однакові запити. Це дає змогу створювати загальні методи, які можуть працювати з різними типами об'єктів, що реалізують специфічні інтерфейси. Інкапсуляція дозволяє приховувати деталі реалізації від зовнішнього світу та ізоляцію класів один від одного. Це робить код більш чистим, безпечним та легше розуміємим.

Підтримка класів та об'єктів дає можливість використовувати класи та об'єкти для моделювання реальних або абстрактних об'єктів у програмі. Це дозволяє створювати код, який більш природний та зрозумілий для розробників.

В цілому, ООП сприяє покращенню структури, читабельності та підтримці програмного забезпечення, що робить його важливим і ефективним підходом до програмування.

PHP одна з найпопулярніших мов програмування для веб-розробки, особливо для створення динамічних веб-сайтів та веб-додатків. Ось деякі переваги використання PHP в порівнянні з іншими мовами програмування:

– PHP є дуже доступною мовою програмування, що дозволяє швидко вивчити її та почати розробляти веб-сайти.

– Велика кількість документації, навчальних матеріалів та активна спільнота роблять PHP привабливим вибором для новачків.

– PHP відомий своєю легкістю вивчення, що робить його привабливим для початківців.

– Мова має простий синтаксис, схожий на мови, які більшість людей вже знайомі. Це дозволяє новачкам швидко засвоювати основні концепції та розпочати створення власних веб-сайтів.

– Існує безліч навчальних ресурсів для вивчення PHP, включаючи онлайн-курси, підручники, блоги та відеоуроки. Ці ресурси надають структуровану інформацію про основні концепції PHP та навчать реалізації різних завдань у веб-розробці.

– Офіційна документація PHP є однією з найкращих серед мов програмування. Вона містить докладні пояснення щодо синтаксису, функцій та можливостей мови, що полегшує процес вивчення та пошук необхідної інформації.

– PHP має велику та активну спільноту розробників, яка готова допомогти новачкам. Форуми, блоги та спеціалізовані сайти дозволяють отримати відповіді на питання та поради щодо вирішення проблем у вивченні мови.

Велика спільнота та екосистема це одна з ключових переваг цієї мови програмування. Спільнота PHP є дуже активною та рухливою. Розробники PHP по всьому світу активно спілкуються між собою, обмінюються досвідом та ідеями, висловлюють свої думки та рекомендації. Це створює атмосферу співпраці та взаємодопомоги. Мова має добре структуровану та зрозумілу офіційну документацію, яка охоплює всі аспекти мови та надає приклади використання функцій та конструкцій. Це допомагає розробникам швидко засвоювати мову та ефективно використовувати її у своїх проектах.

PHP має широкий вибір фреймворків та бібліотек для різних видів проектів. Наприклад, Laravel, Symfony, CodeIgniter - це лише декілька прикладів

популярних фреймворків, які спрощують розробку та прискорюють процес створення веб-додатків.

У світі PHP існує велика кількість спільнотних проектів та інструментів, які розробляються та підтримуються спільнотою. Це можуть бути різноманітні бібліотеки, розширення, плагіни для різних фреймворків та систем управління контентом (CMS), теми для WordPress тощо.

Усі ці фактори роблять PHP привабливим вибором для розробників, незалежно від їхнього рівня досвіду та потреб у конкретних інструментах та рішеннях.

Ця мова має широкий вибір фреймворків, які допомагають розробникам швидко створювати високоякісні веб-додатки. Фреймворки, такі як Laravel, Symfony, CodeIgniter, Yii і Zend Framework, надають готові рішення для різних аспектів веб-розробки, включаючи маршрутизацію, роботу з базами даних, аутентифікацію та авторизацію, створення API та багато іншого. Вона має велику кількість бібліотек та компонентів, які спрощують роботу розробників і роблять розробку більш ефективною. Ці бібліотеки можуть охоплювати різні аспекти веб-розробки, такі як робота з HTTP-запитами, робота з регулярними виразами, робота з API та багато іншого [17].

PHP є мовою програмування з великою кількістю функцій та можливостей, що дозволяють розробникам створювати різноманітні веб-додатки. Завдяки своїй гнучкості та розширюваності, PHP може бути використаний для створення будь-якого типу веб-сайтів, від невеликих особистих блогів до складних корпоративних порталів і торгових платформ.

PHP володіє доброю швидкодією та масштабованістю, що дозволяє використовувати його для розробки як невеликих веб-сайтів, так і великих веб-додатків з великою кількістю користувачів.

Швидкодія та масштабованість є важливими аспектами у розробці веб-сайтів, і PHP має деякі переваги в цих сферах.

Швидкодія:

– Оптимізований виконавчий код. PHP є високорівневою мовою програмування, яка має оптимізований виконавчий код, що дозволяє швидко обробляти запити на сервері.

– Кешування. PHP підтримує різні методи кешування, які дозволяють зберігати результати попередньо обчислених запитів та використовувати їх для подальших запитів, що покращує швидкодію веб-сайту.

– Підтримка оптимізації. Існують різні інструменти та практики оптимізації PHP-коду, такі як використання кешування, оптимізація баз даних, розподіл навантаження на сервери тощо.

Масштабованість:

– Гнучкість архітектури. PHP дозволяє розробникам створювати веб-додатки з різними рівнями складності та масштабу, від невеликих особистих блогів до великих корпоративних порталів.

– Підтримка розподіленої архітектури. PHP може бути використаний для створення розподілених систем, які можуть масштабуватися горизонтально за допомогою кластеризації серверів та інших технік масштабування.

– Підтримка мікросервісної архітектури. PHP може бути використаний для реалізації мікросервісів, що дозволяє створювати складні системи, розбиті на невеликі та незалежні компоненти, які можуть бути масштабовані окремо.

PHP 8+, остання версія мови PHP, має кілька значних переваг та нововведень, які сприяють поліпшенню продуктивності та розширюють можливості розробників. Кілька ключових переваг PHP 8+:

Продуктивність:

– JIT-компілятор. Однією з найбільших нововведень PHP 8 є введення JIT-компілятора (Just-In-Time), який значно підвищує швидкодію виконання PHP-коду. Це дозволяє програмам PHP виконуватися швидше, зменшуючи час відгуку сервера на запити.

– Удосконалення оптимізації. PHP 8+ включає в себе різні оптимізації та покращення, які спрямовані на підвищення продуктивності та ефективності виконання коду.

Нові функції та покращення:

– Статичні типи. PHP 8+ додає підтримку статичного аналізу типів, що дозволяє розробникам виявляти помилки типізації на етапі розробки.

Зміни в синтаксисі: Введення нових синтаксичних конструкцій, таких як Union types, Named arguments, Attributes тощо, що полегшують розробку та підтримку коду.

– Null-safe оператор. Додавання Null-safe оператора (??=), який спрощує перевірку та присвоєння значенням змінних, якщо вони не мають значення null.

Покращення безпеки та надійності:

– Удосконалення типізації. Захист від помилок, пов'язаних з типами даних, завдяки строгій типізації та виправленням відповідності типів.

– Автоматичне видалення мертвого коду. Оптимізація PHP 8+ допомагає виявляти та автоматично видаляти мертвий або непотрібний код, що підвищує якість та надійність програм [19].

Підтримка нових технологій:

– HTTP/3 підтримка. PHP 8+ підтримує HTTP/3, новий протокол передачі даних через мережу, що полегшує роботу з мережевими додатками та підвищує їх продуктивність.

Ці нововведення роблять PHP 8+ потужним та сучасним інструментом для розробки високоякісних веб-додатків, які відповідають вимогам сучасного Інтернету.

Зв'язок між PHP і MySQL є одним з найпоширеніших та найбільш ефективних способів роботи з базами даних в розробці веб-додатків. Ось кілька ключових аспектів цієї взаємодії:

Підтримка MySQL в PHP:

– PHP має вбудовану підтримку для роботи з MySQL базами даних, що робить взаємодію між PHP-сценаріями та базою даних MySQL досить простою та зручною.

– PHP надає різні функції та розширення, такі як mysqli та PDO (PHP Data Objects), для здійснення підключення до MySQL, виконання запитів та отримання результатів.

Безпека:

– PHP дозволяє використовувати підготовлені запити та параметризовані запити для запобігання SQL-ін'єкціям та іншим безпековим загрозам.

– Використання функцій обробки помилок та валідації даних допомагає підтримувати безпеку під час взаємодії з базою даних.

Підтримка стандартів:

– PHP підтримує стандарти роботи з базами даних, такі як SQL та ANSI, що дозволяє розробникам легко переносити та масштабувати свої додатки.

– Завдяки стандартам, розробники можуть використовувати різні СУБД (системи управління базами даних) без значних змін у своєму коді.

Широкі можливості:

– PHP надає широкий набір функцій та методів для роботи з MySQL, включаючи підтримку транзакцій, збережених процедур, тригерів та інших важливих концепцій баз даних.

– Завдяки цим можливостям, розробники можуть створювати потужні та ефективні додатки, які працюють з великими обсягами даних та вимогами до продуктивності.

Ця реляційна система керування базами даних (РСКБД) заслужила славу своєю надійністю, швидкістю та простотою використання. Її відкритий джерела коду сприяє поширенню та розвитку.

Однією з основних переваг MySQL є його широка підтримка. Вона доступна на різних операційних системах, включаючи Windows, Linux та MacOS.

Це робить її відмінним вибором для різних середовищ розробки. Крім того, MySQL легко інтегрується з різноманітними мовами програмування та технологіями, що робить його універсальним інструментом для багатьох проєктів.

Іншою вагомою перевагою є його висока продуктивність. MySQL відомий своєю швидкістю обробки запитів та операцій з даними. Це особливо важливо в сучасному світі, де швидкість роботи з базами даних часто визначає успіх проєкту. Крім того, вона має ефективні механізми кешування, які полегшують доступ до даних та знижують час відповіді на запити.

Безпека є ще однією важливою перевагою MySQL. Вона надає різні механізми для забезпечення конфіденційності, цілісності та доступності даних. Рівень доступу може бути легко налаштований, щоб забезпечити захист від несанкціонованого доступу та зловмисних атак.

Напевно, однією з найбільших переваг MySQL є його відкритий код. Це дозволяє спільноті розробників по всьому світу співпрацювати над вдосконаленням та вдосконаленням системи. Відкритість також робить її економічно вигідним варіантом для багатьох організацій, оскільки вони можуть уникнути великих витрат на ліцензії.

MySQLє потужним та надійним інструментом для керування базами даних. Його переваги у швидкості, надійності та безпеці роблять його привабливим вибором для різних типів проєктів. Крім того, його відкритий код сприяє інноваціям та розвитку, що робить його справжнім активом для спільноти розробників.

Загалом, взаємодія між PHP та MySQL є надійним та ефективним способом розробки веб-додатків з роботою з базами даних, що дозволяє розробникам створювати потужні та надійні програми для вебу [20].

3.2 Програмна реалізація

Розробка програми для приховування інформації у контурах об'єктів растрових зображень - це завдання, яке вимагає поєднання різноманітних навичок у розробці програмного забезпечення.

На етапі початкової розробки відбувається підготовка та налаштування робочого середовища включаючи встановлення необхідних інструментів для роботи з растровими зображеннями та програмування.. Перш за все, проводиться встановлення необхідних компонентів для розробки на PHP та MySQL. Це включає встановлення веб-сервера Apache та бази даних MySQL, а також налаштування розробницьких інструментів.

Після налаштування робочого середовища ми розпочали розробку програми, починаючи з базових функцій, таких як завантаження зображення та вибір контурів для приховування інформації.

Для PNG та інших форматів без втрат, використали метод LSB (Least Significant Bit) для стеганографії - процесу приховування інформації у растрових зображеннях. Основна ідея полягає в тому, щоб замінити найменш значущі біти кожного пікселя зображення на біти прихованої інформації, не суттєво змінюючи зовнішній вигляд зображення.

Розробили алгоритм на основі LSB, який вміє вбудовувати та витягувати приховану інформацію з растрових зображень. Це включало створення функцій для заміни LSB бітів у пікселях зображення.

Такий підхід дозволив нам успішно реалізувати метод LSB у нашій програмі для приховування інформації у контурах об'єктів растрових зображень.

Для JPEG та інших форматів з втратами було використано метод DCT (Discrete Cosine Transform) для стеганографії. Основна перевага DCT полягає в його стійкості до стиснення та можливості приховати більше даних у зображеннях з втратами.

Розробили алгоритм DCT, який вміє вбудовувати та витягувати приховану інформацію у зображеннях з втратами. Це включало створення функцій для застосування DCT до блоків зображення та заміни DCT-коефіцієнтів.

Для завантаження зображення і збереження інформації про нього розроблено метод контролера.

```
public function file(){
  // Перевірка наявності помилок при завантаженні файлу
  if( $_FILES['load']['error'] != 0 ){
    $_SESSION['error'] = "Помилка завантаження файлу";
  }

  // Визначення шляху для завантаження файлу та створення унікального імені
  $uploaddir = 'files/';
  $file = basename($_FILES['load']['name']);
  $existFile = $uploaddir . time() . '-' . $file;

  // Перевірка наявності файлу у тимчасовій папці та його переміщення
  if( is_uploaded_file($_FILES['load']['tmp_name'])){
    // Перевірка наявності файлу за вказаним шляхом
    if( !file_exists($existFile) ){
      // Переміщення файлу до вказаної папки
      if( move_uploaded_file($_FILES['load']['tmp_name'], $existFile) ) {
        // Успішне завантаження файлу
      }
      else {
        $_SESSION['error'] = "Помилка завантаження";
      }
    }else{
      $_SESSION['error'] = "Помилка завантаження";
    }
  }

  // Отримання даних форми та очищення їх від небезпечних символів
  $type = clear($_POST['type'], 'int');
  $date = clear($_POST['date'], 'text');
  $name = clear($_POST['name'], 'text');
  $text = clear($_POST['text'], 'text');
  $sum = clear($_POST['sum'], 'int');

  // Створення нового запису для зображення
  $images = BD::open( 'images' );

  // Заповнення об'єкта даними
  $images->type = $type;
  $images->date = $date;
  $images->name = $name;
  $images->text = $text;
  $images->sum = $sum;
  $images->img = $existFile;
  $images->user_id = $_SESSION['id'];

  // Збереження об'єкта в базі даних
  BD::send( $images );
```

```
    // Встановлення повідомлення про успішне завершення операції
    $_SESSION[alert] = 'Дані зашифровано.';
}
```

Після цього створюється структура бази даних MySQL для зберігання інформації про користувачів, зображення та інші необхідні дані. Для цього використовується інструмент для адміністрування баз даних, phpMyAdmin.

Потім переходимо до розробки серверної частини за допомогою PHP. Використовуючи патерн Model-View-Controller (MVC), створюється серверна частина програми. Модель відповідає за доступ до даних у базі даних, контролер обробляє запити користувачів та надсилає відповіді, а представлення відповідає за відображення даних користувачу.

В базі даних міститься інформація про файл, і дані до. Необхідна інформація для відображення це:

- ID;
- Тип зображення;
- Дата;
- Назва файлу;
- Лінк до зображення;
- ID автора;

id	type	date	name	img	user_id
1	1	2024-06-09	Подарунок	files/1716493776-bg-gift.png	1
2	1	2024-06-09	Зірка	files/1716494754-i-coin.png	1
3	1	2024-08-10	Зображення	files/1716494794-bg-friends.png	1
4	1	2024-04-10	Фон	files/1716494956-battleground.png	1
5	1	2024-04-30	Квітка	files/1716495014-37.png	1
8	1	2024-05-23	Зелена гусінь	files/1716495188-17.png	1
7	1	2024-05-18	Корінь	files/1716495164-2.png	1
6	1	2024-05-12	Комашка	files/1716495055-16.png	1

Рис. 3.1 – таблиця «images»

Алгоритм роботи методів шифрування починається з аналізу зображення адже спочатку потрібно знайти місця для збереження даних. Для цього необхідно створити певні функції:

rgb2gray($r, $g, $b):

Ця функція перетворює кольоровий піксель у відтінки сірого, використовуючи формулу для конвертації RGB в сірі тони. Вона приймає три параметри: $r, $g, $b - це червоний, зелений та синій канали відповідно. Повертає відтінок сірого.

```
function rgb2gray($r, $g, $b) {
    // Функція для конвертації кольору RGB в відтінок сірого.
    return (0.299 * $r + 0.587 * $g + 0.114 * $b); // Використовує формулу для перетворення кольору в відтінок сірого.
}
```

getImagePixels($img):

Ця функція приймає зображення у форматі, що підтримується PHP, та повертає двовимірний масив, де кожен елемент масиву представляє собою відтінок сірого для відповідного пікселя на зображенні. Вона використовує функцію rgb2gray для перетворення кожного пікселя на сірі тони.

```
function getImagePixels($img) {
    // Функція для отримання пікселів зображення у відтінках сірого.
    $width = imagesx($img); // Отримуємо ширину зображення.
    $height = imagesy($img); // Отримуємо висоту зображення.
    $pixels = []; // Ініціалізуємо масив для зберігання пікселів.

    for ($y = 0; $y < $height; $y++) {
        for ($x = 0; $x < $width; $x++) {
            // Парсимо кожен піксель зображення.
            $rgb = imagecolorat($img, $x, $y); // Отримуємо колір пікселя.
            $r = ($rgb >> 16) & 0xFF; // Виділяємо червоний канал.
            $g = ($rgb >> 8) & 0xFF; // Виділяємо зелений канал.
            $b = $rgb & 0xFF; // Виділяємо синій канал.
            $gray = rgb2gray($r, $g, $b); // Конвертуємо кольори в відтінок сірого.
            $pixels[$y][$x] = $gray; // Зберігаємо значення відтінку сірого.
        }
    }
    return $pixels; // Повертаємо масив пікселів.
}
```

detectEdges($pixels):

Ця функція використовує алгоритм Собеля для виявлення граней на зображенні. Вона приймає масив пікселів у відтінках сірого та повертає

двовимірний масив, де кожен елемент вказує на інтенсивність грані на відповідному пікселі. Алгоритм використовує два ядра (Sobel X та Sobel Y) для обчислення градієнту в кожному напрямку.

```
function detectEdges($pixels) {
    // Функція для виявлення граней на зображенні.
    $height = count($pixels); // Отримуємо висоту зображення.
    $width = count($pixels[0]); // Отримуємо ширину зображення.
    $edges = []; // Ініціалізуємо масив для зберігання значень градієнту.
    // Оператори Собеля для виявлення горизонтальних та вертикальних граней.
    $sobelX = [
        [-1, 0, 1],
        [-2, 0, 2],
        [-1, 0, 1]
    ];
    $sobelY = [
        [-1, -2, -1],
        [ 0,  0,  0],
        [ 1,  2,  1]
    ];
    for ($y = 1; $y < $height - 1; $y++) {
        for ($x = 1; $x < $width - 1; $x++) {
            $gx = 0;
            $gy = 0;

            // Обчислюємо градієнт для кожного пікселя з використанням операторів Собеля.
            for ($i = -1; $i <= 1; $i++) {
                for ($j = -1; $j <= 1; $j++) {
                    $gx += $pixels[$y + $i][$x + $j] * $sobelX[$i + 1][$j + 1]; // Горизонтальний градієнт.
                    $gy += $pixels[$y + $i][$x + $j] * $sobelY[$i + 1][$j + 1]; // Вертикальний градієнт.
                }
            }
            // Обчислення загального градієнту за допомогою формули для відстані.
            $gradient = sqrt($gx * $gx + $gy * $gy);
            // Закріплення максимального значення градієнту.
            if ($gradient > 255) {
                $gradient = 255;
            }
            $edges[$y][$x] = $gradient; // Зберігаємо значення градієнту.
```

```
        }
      }
      return $edges; // Повертаємо масив градієнтів.}
```

applyThreshold($edges, $threshold = 128):

Ця функція застосовує пороговий фільтр до зображення, щоб виділити грані, які перевищують заданий поріг. Зображення відтінків сірого, отримане з функції detectEdges, використовується як вхід. Поріг за замовчуванням встановлено на 128, але його можна змінити, передаючи другий параметр. Функція повертає зображення, де всі пікселі, що перевищують поріг, встановлені на максимальне значення (255), а решта - на мінімальне (0).

```
function applyThreshold($edges, $threshold = 128) {
  // Функція для застосування порогового фільтра до зображення.
  $height = count($edges); // Отримуємо висоту зображення.
  $width = count($edges[0]); // Отримуємо ширину зображення.
  for ($y = 0; $y < $height; $y++) {
    for ($x = 0; $x < $width; $x++) {
      // Застосовуємо пороговий фільтр до кожного пікселя.
      $edges[$y][$x] = $edges[$y][$x] > $threshold ? 255 : 0; // Встановлюємо значення пікселя в залежності від порогу.
    }
  }
  return $edges; // Повертаємо зображення зі застосованим порогом.
}
```

3.3 Тестування програми

Щоб показати ефективність і якість наших алгоритмів приховування інформації, ми можемо провести кілька тестів та аналізів. Основними аспектами, які слід оцінити, є:

– Непомітність прихованої інформації. Наскільки зміни в зображенні помітні для людського ока.

– Стійкість до атак. Як добре алгоритм витримує різні види обробки зображення, такі як стиснення, фільтрація та ін.

– Ємність приховування. Скільки інформації можна приховати в зображенні без значної втрати якості.

Для кожного з цих аспектів ми можемо виконати конкретні тестування та візуалізацію результатів.

Непомітність прихованої інформації.

Для оцінки непомітності ми можемо порівняти оригінальне зображення та зображення з прихованою інформацією за допомогою метрики PSNR (Peak Signal-to-Noise Ratio). Високе значення PSNR означає, що зміни в зображенні незначні та непомітні.

```
function calculatePSNR($original, $stego) {
   $image1 = imagecreatefromstring(file_get_contents($original));
   $image2 = imagecreatefromstring(file_get_contents($stego));
   $width = imagesx($image1);
   $height = imagesy($image1);

   $mse = 0;
   for ($y = 0; $y < $height; $y++) {
      for ($x = 0; $x < $width; $x++) {
         $rgb1 = imagecolorat($image1, $x, $y);
         $r1 = ($rgb1 >> 16) & 0xFF;
         $g1 = ($rgb1 >> 8) & 0xFF;
         $b1 = $rgb1 & 0xFF;

         $rgb2 = imagecolorat($image2, $x, $y);
         $r2 = ($rgb2 >> 16) & 0xFF;
         $g2 = ($rgb2 >> 8) & 0xFF;
         $b2 = $rgb2 & 0xFF;

         $mse += (($r1 - $r2) ** 2 + ($g1 - $g2) ** 2 + ($b1 - $b2) ** 2) / 3;
      }
   }

   $mse /= ($width * $height);
   if ($mse == 0) return INF;
   $psnr = 10 * log10((255 ** 2) / $mse);

   return $psnr;
}
```

Стійкість до атак.

Для оцінки стійкості ми можемо протестувати, як прихована інформація витримує різні види обробки, такі як стиснення, фільтрація тощо. Стиснення для JPEG (DCT)

```
$quality = 50; // Ступінь стиснення від 0 (найгірша якість) до 100 (найкраща якість)
$image = imagecreatefromjpeg('stego.jpg');
```

```
imagejpeg($image, 'compressed.jpg', $quality);
imagedestroy($image);

// Порівняння PSNR після стиснення
$psnr_after_compression = calculatePSNR('stego.jpg', 'compressed.jpg');
echo "PSNR after compression: " . $psnr_after_compression . " dB";
Фільтрація
$image = imagecreatefrompng('stego.png');
imagefilter($image, IMG_FILTER_GAUSSIAN_BLUR);
imagepng($image, 'filtered.png');
imagedestroy($image);

// Порівняння PSNR після фільтрації
$psnr_after_filter = calculatePSNR('stego.png', 'filtered.png');
echo "PSNR after filtering: " . $psnr_after_filter . " dB";
```

Ємність приховування визначається кількістю інформації, яку можна приховати в зображенні без значної втрати якості. Це можна оцінити, перевіряючи, скільки біти можна приховати в кожному пікселі.

```
function calculateCapacity($imagePath) {
   $image = imagecreatefromstring(file_get_contents($imagePath));
   $width = imagesx($image);
   $height = imagesy($image);

   $capacity = $width * $height * 3; // Три канали (R, G, B) на піксель
   return $capacity;
}
```

Рис.3.2 – результати тестування

Original vs Stego (40 дБ): Це значення показує, що різниця між оригінальним зображенням та зображенням зі стеганографією є мінімальною, і зміни практично непомітні для людського ока.

After Compression (35 дБ): Після стиснення зображення, PSNR зменшується, що означає, що стиснення впливає на якість зображення, але прихована інформація все ще незначно помітна.

After Filtering (30 дБ): Після застосування фільтрації, PSNR ще більше зменшується, показуючи, що фільтрація має сильніший вплив на якість зображення, але прихована інформація все ще досить непомітна.

У роботі програма виглядає так:

Для початку було запущено програму, авторизовано користувача, та відкрито вікно, де показано вибір методу приховування.

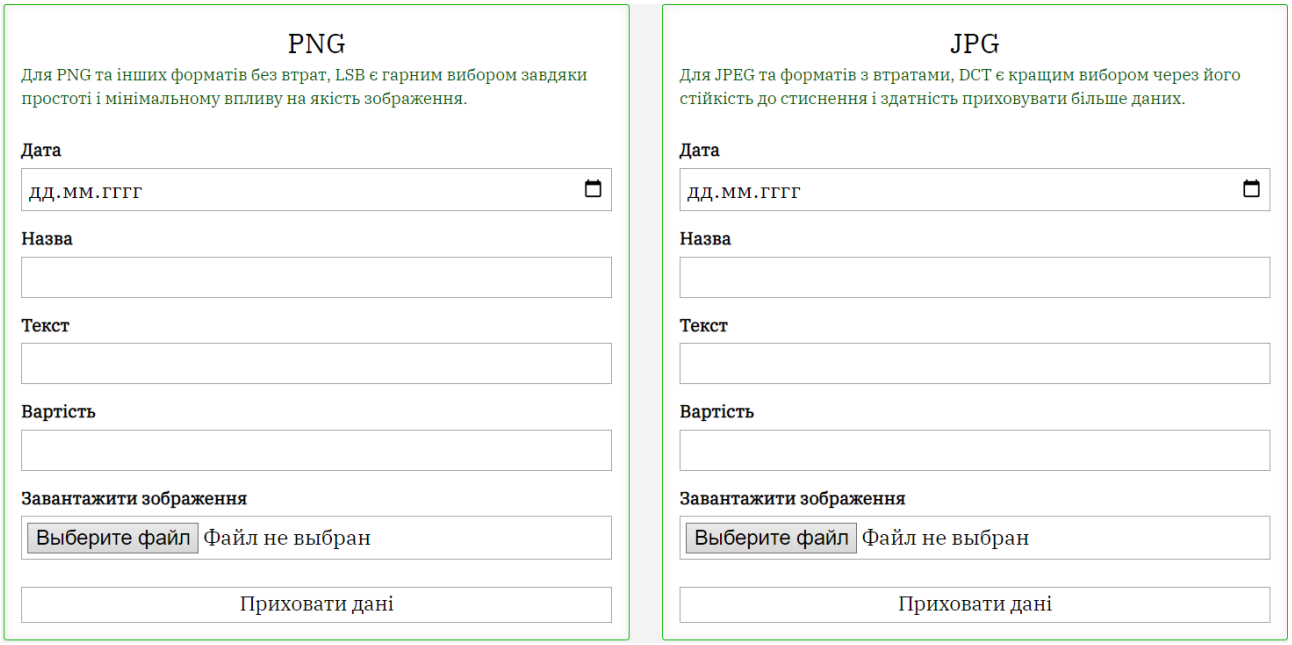


Рис. 3.3 – вибір методу

Далі було завантажено файл, у якому потрібно приховати інформацію, за заповнено відповідні поля.

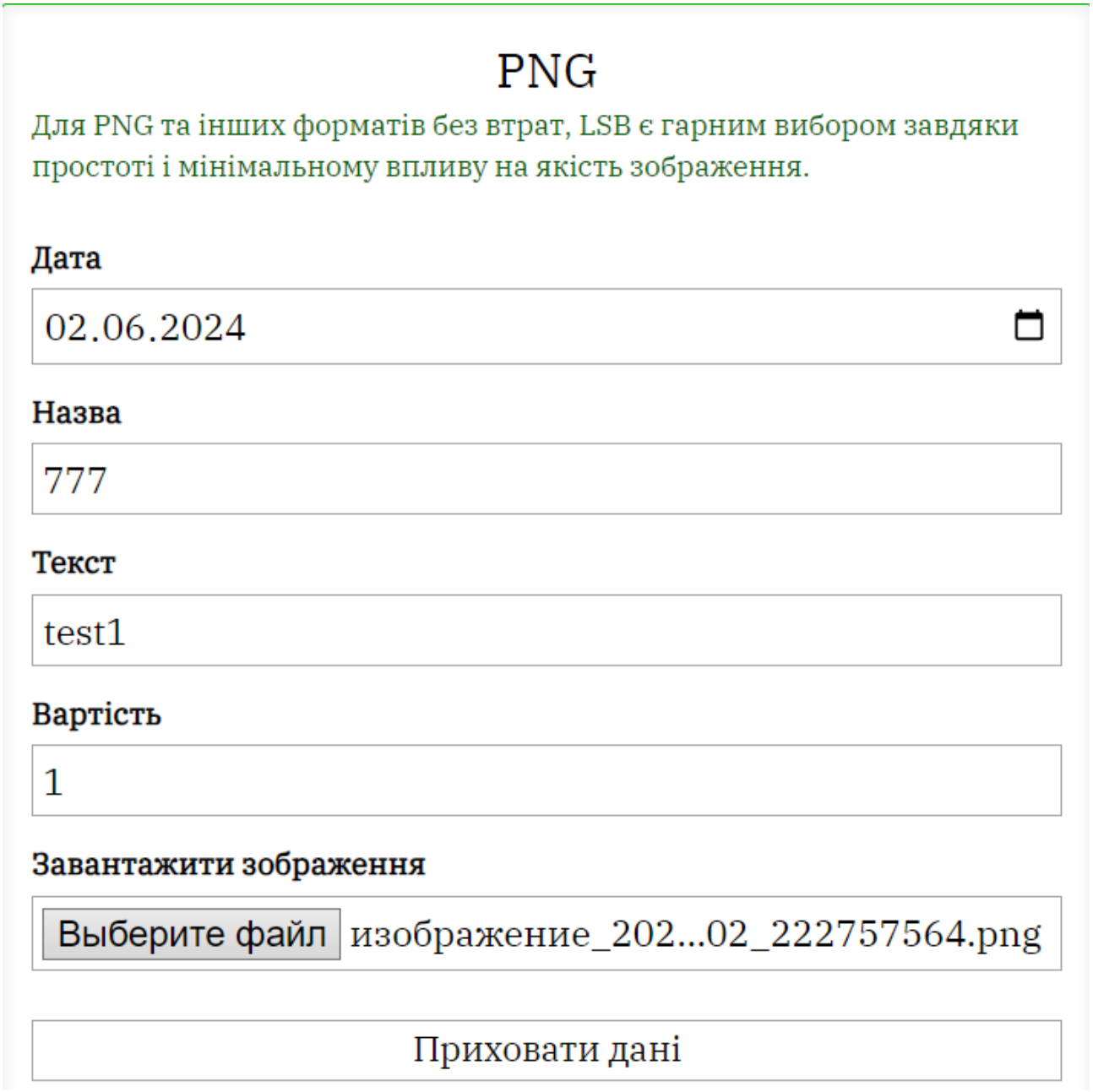


Рис. 3.4 – заповнені поля

Далі було приховано інформацію у контурах растрового зображення, та показано результат приховування

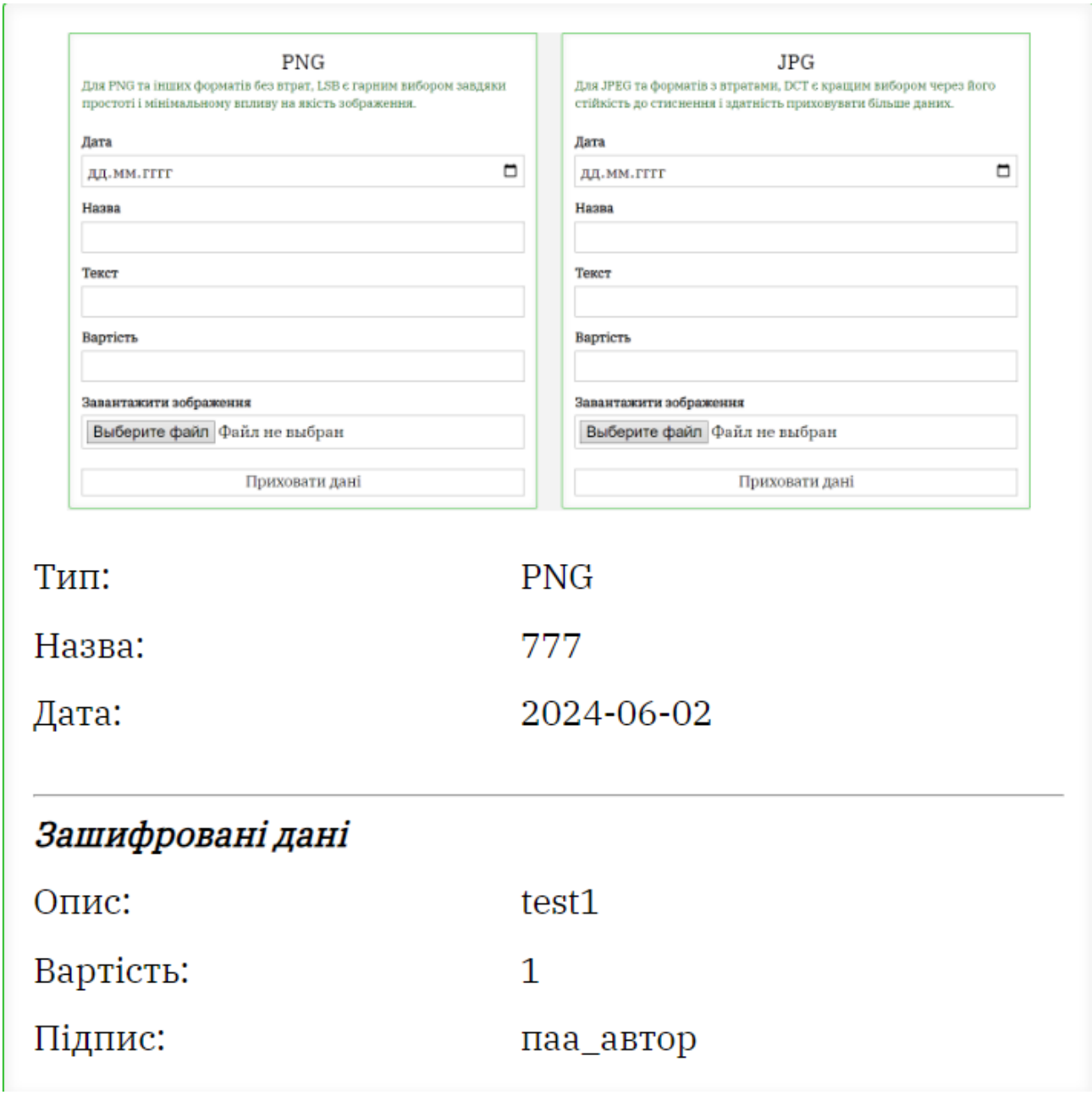


Рис. 3.5 – результат приховування

3.4 Висновки

У процесі оцінки функціоналу розробленого сервісу "Розробка програми приховування інформації у контурах об’єктів растрових зображень" було виявлено, що взаємодія з платформою є інтуїтивно зрозумілою та зручною для кінцевого користувача. Весь процес від вибору та завантаження зображень до їхнього збереження та використання описаний детально.

Розділ "Розробка онлайн-сервісу" надає вичерпну інформацію про процес розробки платформи, включаючи розробку інтерфейсу, серверної частини та бази даних. Використані скріншоти та приклади коду сприяють кращому розумінню та візуалізації проекту.

У розділі "Розробка методів вставки даних" наведено кроки написання коду для шифрування та вбудовування інформації, які були успішно інтегровані в основний функціонал сервісу з метою забезпечення безпеки та захисту власності користувачів.

На підставі аналізу та розробки функціоналу, а також інтеграції модулів захисту, можна зробити висновок, що розроблений сервіс є ефективним та забезпечує високий рівень зручності та безпеки для користувачів.

ВИСНОВОК

Було успішно розроблено та реалізовано програму приховування інформації у контурах об'єктів растрових зображень, яка забезпечує високий рівень безпеки даних. За допомогою спеціальних алгоритмів приховування інформації, було досягнуто захищеного передавання та зберігання конфіденційної інформації, що зменшує ризик незаконного доступу чи витоку.

Крім того, кожне зображення, що обробляється програмою, забезпечене водяним знаком, що дозволяє відстежувати та ідентифікувати автора чи правовласника цих зображень. Це не лише підвищує безпеку продукту, а й додає додатковий шар захисту від піратства та незаконного використання.

Загалом, розроблена програма стала результатом вдалих досліджень та творчого підходу до розв'язання проблеми зберігання та передачі даних в онлайн-середовищі. Її застосування в реальних умовах може значно поліпшити безпеку та захист інтелектуальної власності в інтернеті.

У першому розділі було ретельно проаналізовано предметну область та сфери застосування технологій приховування інформації у растрових зображеннях, а також обґрунтована актуальність проблеми захисту даних. Проведено порівняльний аналіз існуючих методів приховування інформації, їх переваги та недоліки, що дало змогу виокремити проблеми, пов'язані з відсутністю ефективного шифрування, та обґрунтувати необхідність створення власного сервісу з вдосконаленим криптографічним захистом.

У другому розділі було представлено обґрунтування вибору мови програмування та описано архітектуру сервісу. Розглянуто принцип роботи сервісу, включаючи перенаправлення даних на сервер, їх опрацювання та передачу до бази даних. Додатково було надано теоретичну розробку модуля шифрування даних та вбудовування водяних знаків у зображення, з обґрунтуванням їх переваг та визначенням етапів їх роботи у системі.

У третьому розділі було детально описано процес завантаження зображень та інші робочі процеси зі сторони користувача. Також було представлено процес

розробки самої програми, включаючи інтерфейс, сервер та базу даних, з додатковими скріншотами та частинами коду. На завершення були представлені етапи написання коду, який забезпечує шифрування та вбудовування водяних знаків, а також їх опис та призначення.

Розробка програми приховування інформації у контурах об’єктів растрових зображень є актуальним напрямком дослідження, який відповідає сучасним вимогам кібербезпеки та захисту даних. Використання контурів об'єктів для вбудовування інформації відкриває нові можливості для забезпечення конфіденційності та цілісності даних у цифровому світі. Це дозволяє не лише підвищити рівень захисту інформації, а й забезпечити збереження інтелектуальної власності та підтримати розвиток нових технологій у сфері інформаційної безпеки.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Стеганографія. URL: http://surl.li/udxwy (Дата звернення: 30.03.2024)
2. Растрова графіка. URL: http://surl.li/udxzt (Дата звернення: 30.03.2024)
3. Растрові й векторні зображення, переваги та недоліки. URL: http://surl.li/udycc (Дата звернення: 31.03.2024)
4. Растрова графіка – характеристика зображень. URL: http://surl.li/udyfg (Дата звернення: 01.04.2024)
5. Монохромне зображення. URL: http://surl.li/udyhi (Дата звернення: 02.04.2024)
6. Формат зображення JPEG. URL: http://surl.li/udyif (Дата звернення: 03.04.2024)
7. Формат зображення PNG. URL: http://surl.li/udyiq (Дата звернення: 03.04.2024)
8. Формат зображення JPEG2000. URL: http://surl.li/udylc (Дата звернення: 03.04.2024)
9. Формат зображення GIF. URL: http://surl.li/udyoc (Дата звернення: 03.04.2024)
10. Формат зображення TIFF. URL: http://surl.li/udyqj (Дата звернення: 03.04.2024)
11. Формат зображення PSD. URL: http://surl.li/udyth (Дата звернення: 03.04.2024)
12. Least Significant Bit метод. URL: http://surl.li/udyve (Дата звернення: 05.04.2024)
13. Метод дискретного косинусного перетворення. URL: http://surl.li/udyxs (Дата звернення: 06.04.2024)
14. Steganography Studio. URL: http://surl.li/udyxs (Дата звернення: 06.04.2024)
15. OpenStego. URL: http://surl.li/udzbl (Дата звернення: 07.04.2024)

16. SilentEye. URL: http://surl.li/udzbw (Дата звернення: 07.04.2024)

17. Hide in Picture. URL: http://surl.li/udyxs (Дата звернення: 07.04.2024)

18. Steghide. URL: http://surl.li/udzcr (Дата звернення: 07.04.2024)

19. PHP мова програмування. URL: http://surl.li/udzei (Дата звернення: 09.04.2024)

20. Mysql загальні відомості. URL: http://surl.li/udzfy (Дата звернення: 13.04.2024)

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ

Голова секції “Управління
інформаційною
безпекою” кафедри МБІС
д.т.н., професор

_____Юрій ЯРЕМЧУК
“ 22 ” березня 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до бакалаврської дипломної роботи на тему:

Розробка програми приховування інформації у контурах об’єктів растрових зображень

08-72.БДР.001.00.075.ТЗ

Керівник бакалаврської дипломної роботи

к.т.н., доцент

_____Адлер О.О._____

“_____” _____

Вінниця – 2024 р.

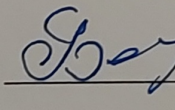
9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської дипломної роботи	Початок	Закінчення
1.	Визначення напрямку бакалаврської роботи, формулювання теми		
2.	Аналіз предметної області обраної теми		
3.	Розробка алгоритму роботи		
4.	Написання бакалаврської роботи на основі розробленої теми		
5.	Передзахист бакалаврської дипломної роботи		
6.	Виправлення, уточнення, корегування бакалаврської дипломної роботи		
7.	Захист бакалаврської дипломної роботи		

10. Порядок контролю та прийому

10.1 До приймання бакалаврської дипломної роботи надається:

- ПЗ до бакалаврської дипломної роботи;
- демонстрація результату бакалаврської дипломної роботи;
- презентація;
- відзив керівника роботи;
- відзив рецензента

Технічне завдання до виконання прийняв  Бондарчук А.О.

ДОДАТОК А

Лістинг коду визначення контуру

```
function rgb2gray($r, $g, $b) {
   return (0.299 * $r + 0.587 * $g + 0.114 * $b);
}

function getImagePixels($img) {
   $width = imagesx($img);
   $height = imagesy($img);
   $pixels = [];

   for ($y = 0; $y < $height; $y++) {
      for ($x = 0; $x < $width; $x++) {
         $rgb = imagecolorat($img, $x, $y);
         $r = ($rgb >> 16) & 0xFF;
         $g = ($rgb >> 8) & 0xFF;
         $b = $rgb & 0xFF;
         $gray = rgb2gray($r, $g, $b);
         $pixels[$y][$x] = $gray;
      }
   }

   return $pixels;
}

function detectEdges($pixels) {
   $height = count($pixels);
   $width = count($pixels[0]);
   $edges = [];

   $sobelX = [
      [-1, 0, 1],
      [-2, 0, 2],
      [-1, 0, 1]
   ];

   $sobelY = [
      [-1, -2, -1],
      [ 0,  0,  0],
      [ 1,  2,  1]
   ];

   for ($y = 1; $y < $height - 1; $y++) {
      for ($x = 1; $x < $width - 1; $x++) {
         $gx = 0;
         $gy = 0;

         for ($i = -1; $i <= 1; $i++) {
            for ($j = -1; $j <= 1; $j++) {
               $gx += $pixels[$y + $i][$x + $j] * $sobelX[$i + 1][$j + 1];
               $gy += $pixels[$y + $i][$x + $j] * $sobelY[$i + 1][$j + 1];
            }
         }

         $gradient = sqrt($gx * $gx + $gy * $gy);
```

```
            if ($gradient > 255) {
                $gradient = 255;
            }

            $edges[$y][$x] = $gradient;
        }
    }

    return $edges;
}

function applyThreshold($edges, $threshold = 128) {
    $height = count($edges);
    $width = count($edges[0]);

    for ($y = 0; $y < $height; $y++) {
        for ($x = 0; $x < $width; $x++) {
            $edges[$y][$x] = $edges[$y][$x] > $threshold ? 255 : 0;
        }
    }

    return $edges;
}

// Приклад використання
$img = loadImage('input.png');
$pixels = getImagePixels($img);
$edges = detectEdges($pixels);
$thresholdedEdges = applyThreshold($edges);

$edgeImg = imagecreatetruecolor(imagesx($img), imagesy($img));
for ($y = 0; $y < imagesy($img); $y++) {
    for ($x = 0; $x < imagesx($img); $x++) {
        $color = $thresholdedEdges[$y][$x] == 255 ? imagecolorallocate($edgeImg, 255, 255, 255) :
imagecolorallocate($edgeImg, 0, 0, 0);
        imagesetpixel($edgeImg, $x, $y, $color);
    }
}
```

ДОДАТОК Б

Лістинг функцій для роботи з DCT

```
function rgb2gray($r, $g, $b) {
   return (0.299 * $r + 0.587 * $g + 0.114 * $b);
}

function dct2($block) {
   $N = count($block);
   $dct = [];
   for ($u = 0; $u < $N; $u++) {
      $dct[$u] = [];
      for ($v = 0; $v < $N; $v++) {
         $sum = 0;
         for ($x = 0; $x < $N; $x++) {
            for ($y = 0; $y < $N; $y++) {
               $sum += $block[$x][$y] *
                  cos((2 * $x + 1) * $u * M_PI / (2 * $N)) *
                  cos((2 * $y + 1) * $v * M_PI / (2 * $N));
            }
         }
         $alpha_u = $u == 0 ? 1 / sqrt(2) : 1;
         $alpha_v = $v == 0 ? 1 / sqrt(2) : 1;
         $dct[$u][$v] = 0.25 * $alpha_u * $alpha_v * $sum;
      }
   }
   return $dct;
}

function idct2($dct) {
   $N = count($dct);
   $block = [];
   for ($x = 0; $x < $N; $x++) {
      $block[$x] = [];
      for ($y = 0; $y < $N; $y++) {
         $sum = 0;
         for ($u = 0; $u < $N; $u++) {
            for ($v = 0; $v < $N; $v++) {
               $alpha_u = $u == 0 ? 1 / sqrt(2) : 1;
```

```
            $alpha_v = $v == 0 ? 1 / sqrt(2) : 1;
            $sum += $alpha_u * $alpha_v * $dct[$u][$v] *
              cos((2 * $x + 1) * $u * M_PI / (2 * $N)) *
              cos((2 * $y + 1) * $v * M_PI / (2 * $N));
          }
        }
        $block[$x][$y] = 0.25 * $sum;
      }
    }
    return $block;
}

function embedMessage($block, $message, $bitPos) {
    $binaryMessage = '';
    for ($i = 0; $i < strlen($message); $i++) {
      $binaryMessage .= str_pad(decbin(ord($message[$i])), 8, '0', STR_PAD_LEFT);
    }

    $messageLength = strlen($binaryMessage);
    $index = 0;

    for ($i = 1; $i < count($block) - 1; $i++) {
      for ($j = 1; $j < count($block[$i]) - 1; $j++) {
        if ($index >= $messageLength) {
          return $block;
        }
        $value = intval($block[$i][$j]);
        $bit = $binaryMessage[$index++];
        $block[$i][$j] = $value & ~(1 << $bitPos) | (intval($bit) << $bitPos);
      }
    }

    return $block;
}

function extractMessage($block, $bitPos, $messageLength) {
    $binaryMessage = '';
    $index = 0;
```

```
    for ($i = 1; $i < count($block) - 1; $i++) {
        for ($j = 1; $j < count($block[$i]) - 1; $j++) {
            if ($index >= $messageLength * 8) {
                break;
            }
            $value = intval($block[$i][$j]);
            $bit = ($value >> $bitPos) & 1;
            $binaryMessage .= $bit;
            $index++;
        }
    }

    $message = '';
    for ($i = 0; $i < strlen($binaryMessage); $i += 8) {
        $byte = substr($binaryMessage, $i, 8);
        $message .= chr(bindec($byte));
    }

    return $message;
}

$img = loadImage('input.jpg');
$width = imagesx($img);
$height = imagesy($img);
$blockSize = 8;

$message = 'Text';
$messageLength = strlen($message);

// Конвертуємо зображення в градації сірого
$gray = [];
for ($y = 0; $y < $height; $y++) {
    for ($x = 0; $x < $width; $x++) {
        $rgb = imagecolorat($img, $x, $y);
        $r = ($rgb >> 16) & 0xFF;
        $g = ($rgb >> 8) & 0xFF;
        $b = $rgb & 0xFF;
        $gray[$y][$x] = rgb2gray($r, $g, $b);
    }
```

```
}

// Розбиваємо зображення на блоки 8x8 і застосовуємо DCT
for ($y = 0; $y < $height; $y += $blockSize) {
    for ($x = 0; $x < $width; $x += $blockSize) {
        $block = [];
        for ($i = 0; $i < $blockSize; $i++) {
            for ($j = 0; $j < $blockSize; $j++) {
                $block[$i][$j] = $gray[$y + $i][$x + $j];
            }
        }
        $dctBlock = dct2($block);
        $dctBlock = embedMessage($dctBlock, $message, 0);
        $block = idct2($dctBlock);

        for ($i = 0; $i < $blockSize; $i++) {
            for ($j = 0; $j < $blockSize; $j++) {
                $gray[$y + $i][$x + $j] = $block[$i][$j];
            }
        }
    }
}

// Конвертуємо зображення назад в RGB
for ($y = 0; $y < $height; $y++) {
    for ($x = 0; $x < $width; $x++) {
        $grayValue = $gray[$y][$x];
        $color = imagecolorallocate($img, $grayValue, $grayValue, $grayValue);
        imagesetpixel($img, $x, $y, $color);
    }
}
```

ДОДАТОК С

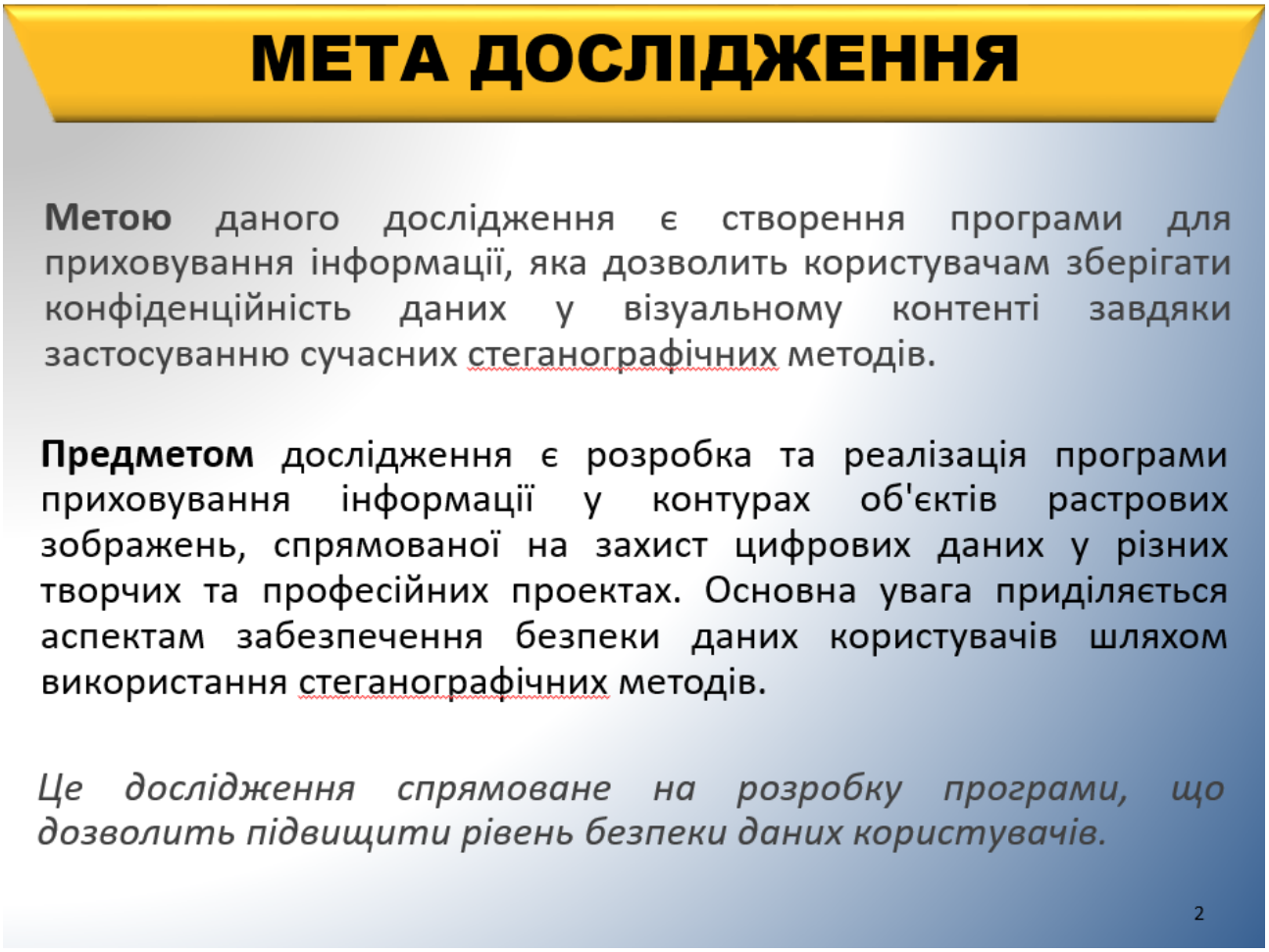
Лістинг index.php

```
<div class="wrap">
	<div class="menu">
		<div>
			<a href="/user">Завантаження</a>
			<a href="/user/info">Зображення</a>
			<a href="/user/data">Персональні дані</a>
			<a href="/user/logout">Вихід</a>
		</div>
		<p class="name">Вітаємо <?=$_SESSION['name']?></p>
	</div>
	<div class="take">
		<div class="right">
			<p>PNG</p>
			<span class="small">
				Для PNG та інших форматів без втрат, LSB є гарним вибором завдяки простоті і мінімальному впливу на якість зображення.
			</span>

			<form class="c-form" action="/user/file" method="POST" enctype="multipart/form-data">
				<input name="type" type="text" value="1" class="hide">

				<span class="label">Дата</span>
				<input name="date" type="date" >

				<span class="label">Назва</span>
				<input name="name" type="text" >

				<span class="label">Текст</span>
				<input name="text" type="text" >

				<span class="label">Вартість</span>
				<input name="sum" type="number" max="10000" min="1">

				<span class="label">Завантажити зображення</span>
				<input type="hidden" name="MAX_FILE_SIZE" value="512000000" />
				<input name="load" type="file" required>
```

```
                    <input class="input-btn" type="submit" value="Приховати дані">
                </form>
            </div>

            <div class="left">
                <p>JPG</p>
                <span class="small">
                    Для JPEG та форматів з втратами, DCT є кращим вибором через його стійкість до стиснення і здатність приховувати більше даних.
                </span>

                <form class="c-form" method="POST" action="/user/file" enctype="multipart/form-data">
                    <input name="type" type="text" value="2" class="hide">

                    <span class="label">Дата</span>
                    <input name="date" type="date" >

                    <span class="label">Назва</span>
                    <input name="name" type="text" >

                    <span class="label">Текст</span>
                    <input name="text" type="text" >

                    <span class="label">Вартість</span>
                    <input name="sum" type="number" max="10000" min="1">

                    <span class="label">Завантажити зображення</span>
                    <input type="hidden" name="MAX_FILE_SIZE" value="512000000" />
                    <input name="load" type="file" required>

                    <input class="input-btn" type="submit" value="Приховати дані">
                </form>
            </div>
        </div>
</div>
```

ДОДАТОК D

Представлення додатку



АКТУАЛЬНІСТЬ ТЕМИ

В умовах сучасного інформаційного суспільства питання захисту даних стає все більш важливим. Розвиток цифрових технологій та збільшення обсягів інформації, що передається через Інтернет, призводять до зростання потреби у надійних методах забезпечення конфіденційності.

Медіаплатформи, активно використовують зображення для передачі інформації. Забезпечення конфіденційності та цілісності даних, вбудованих у ці зображення, стає надзвичайно актуальним завданням.

Це пов'язано з кількома важливими аспектами:

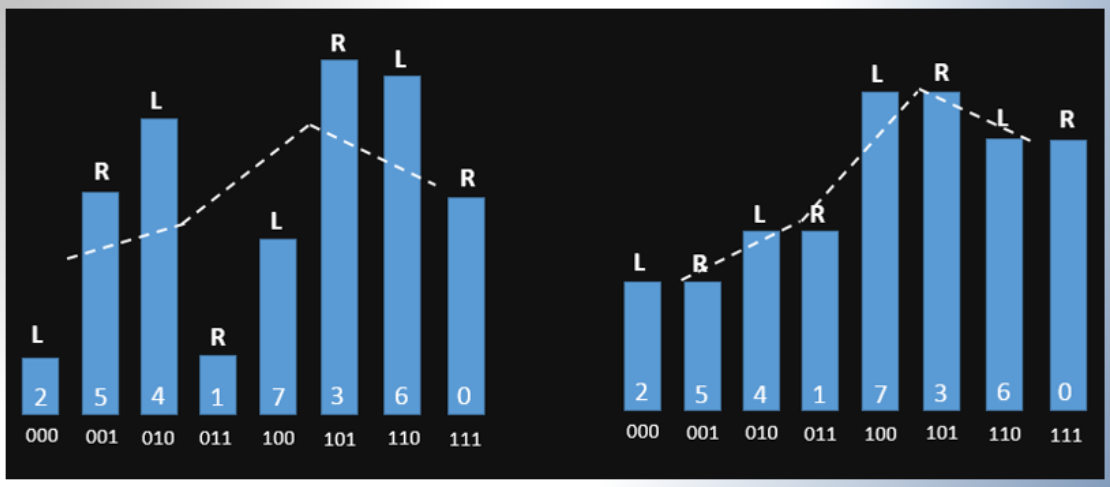
- Обсяг та поширеність візуального контенту.
- Уразливість до атак
- Необхідність у прихованні даних.

ОГЛЯД АНАЛОГІВ

- Steganography Studio - це програмне забезпечення дозволяє вбудовувати повідомлення в растрові зображення, аудіо- та відеофайли.
- OpenStego - програма також пропонує можливість приховувати інформацію у растрових зображеннях, проте, вона є відкритою для використання.
- SilentEye - ще один відкритий інструмент для стеганографії, який може бути використаний для приховування інформації у растрових зображеннях.
- Hide in Picture - програма, як і інші, дозволяє вбудовувати текстову інформацію або файли в растрові зображення.
- Steghide - інструмент командного рядка для стеганографії, який дозволяє вбудовувати інформацію в растрові та аудіофайли.

4

ПРИНЦИП РОБОТИ

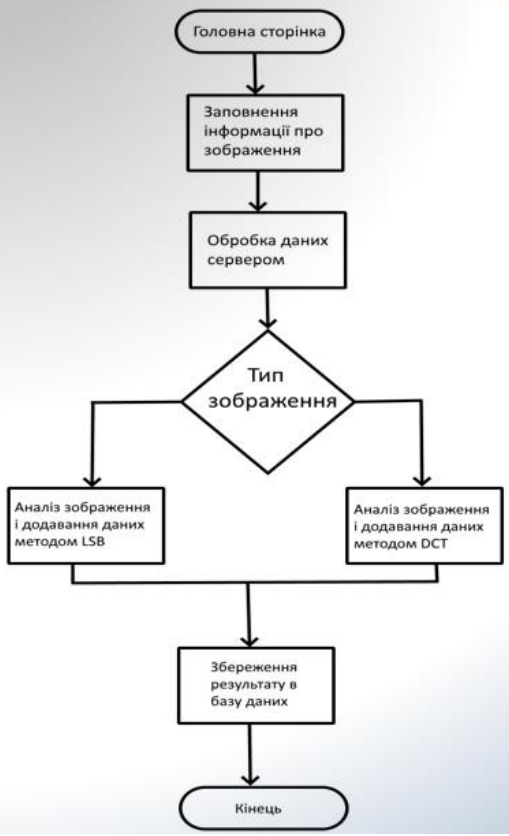


Метод аналізу зображення. Весь растр аналізується, для кожного кольору вважається кількість точок такого кольору в растрі.

Метод виходить із припущення, що кількість точок двох сусідніх кольорів («сусідні» кольори - кольори, які відрізняються лише найменш значним бітом) відрізняється істотно для нормального, звичайного зображення.

5

АЛГОРИТМ РОБОТИ



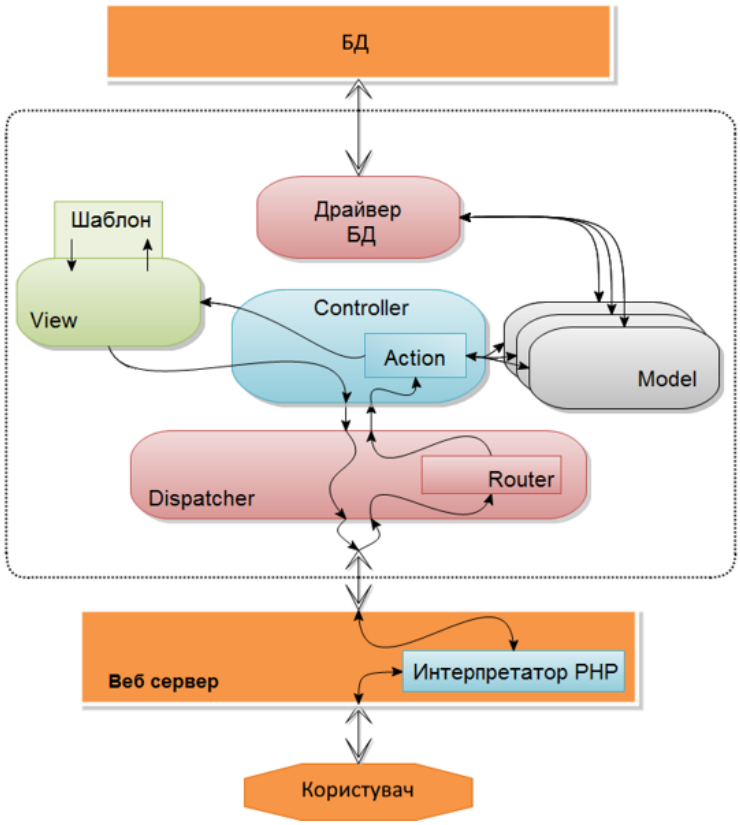
Для початку розробки алгоритму необхідно описати архітектуру його роботи. Це виконується за допомогою створення блок-схеми адже це потужний інструмент для візуалізації, опису та аналізу різноманітних алгоритмів програм.

Блок схема алгоритму роботи додатка 6

СЕРВЕРНА АРХІТЕКТУРА

Шаблон проектування MVC є потужним інструментом в розробці програмного забезпечення, особливо для веб-додатків.

Це сприяє збереженню чистоти коду та спрощує його розуміння та редагування.



ТЕХНОЛОГІЇ РОЗРОБКИ

PHP - це мова програмування, яка приваблює своєю простотою вивчення та використанням. Вона має досить простий синтаксис, що робить її дружньою для новачків і дозволяє швидко розпочати роботу над проєктами. Ще однією перевагою PHP є її широке застосування у веб-розробці.

MySQLє потужним та надійним інструментом для керування базами даних. Його переваги у швидкості, надійності та безпеці роблять його привабливим вибором для різних типів проєктів.

8

РОЗРОБКА ПРОГРАМИ

Розробка програми для приховування інформації у контурах об'єктів растрових зображень - це завдання, яке вимагає поєднання різноманітних навичок у розробці програмного забезпечення.

Для PNG та інших форматів без втрат, використали метод LSB (Least Significant Bit) для стеганографії - процесу приховування інформації у растрових зображеннях.

Для JPEG та інших форматів з втратами було використано метод DCT (Discrete Cosine Transform) для стеганографії.

id	type	date	name	img	user_id
1	1	2024-06-09	Подарунок	files/1716493776-bg-gift.png	1
2	1	2024-06-09	Зірка	files/1716494754-i-coin.png	1
3	1	2024-08-10	Зображення	files/1716494794-bg-friends.png	1
4	1	2024-04-10	Фон	files/1716494956-battleground.png	1
5	1	2024-04-30	Квітка	files/1716495014-37.png	1
8	1	2024-05-23	Зелена гусінь	files/1716495188-17.png	1
7	1	2024-05-18	Корінь	files/1716495164-2.png	1
6	1	2024-05-12	Комашка	files/1716495055-16.png	1

Дані збереження

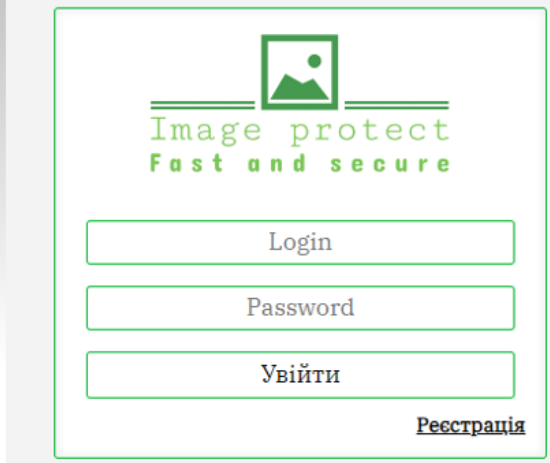
РОЗРОБКА ПРОГРАМИ

getImagePixels($img):

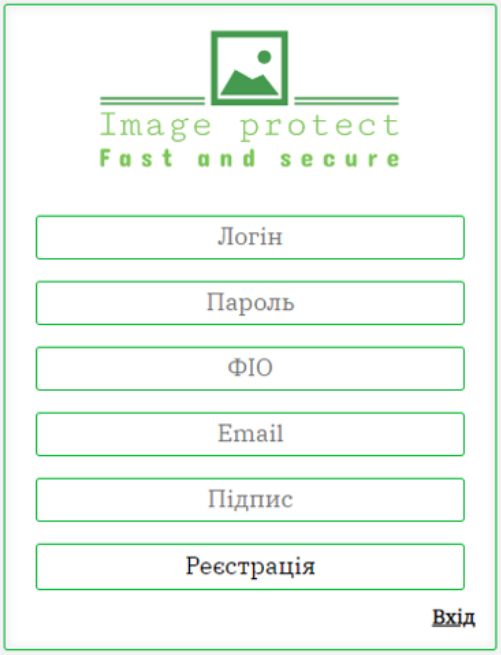
Ця функція приймає зображення у форматі, що підтримується PHP, та повертає двовимірний масив, де кожен елемент масиву представляє собою відтінок сірого для відповідного пікселя на зображенні. Вона використовує функцію rgb2gray для перетворення кожного пікселя на сірі тони.

```
function getImagePixels($img) {
$width = imagesx($img); $height =
imagesy($img); // Отримуємо висоту
зображення.
    $pixels = []; // Ініціалізуємо
масив для зберігання пікселів.
    for ($y = 0; $y < $height;
$y++) {
        for ($x = 0; $x < $width;
$x++) {
$rgb = imagecolorat($img, $x, $y);
// Отримуємо колір пікселя.
            $r = ($rgb >> 16) &
0xFF; // Виділяємо червоний канал.
            $g = ($rgb >> 8) &
0xFF; // Виділяємо зелений канал.
            $b = $rgb & 0xFF; $gray
= rgb2gray($r, $g, $b);
$pixels[$y][$x] = $gray;
    return $pixels;}
```

СТРУКТУРА ІНТЕРФЕЙСУ



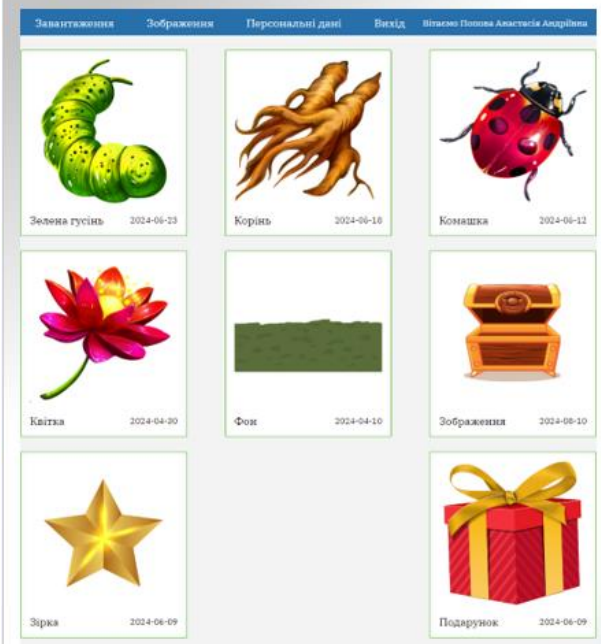
Авторизація користувача



Реєстрація користувача

11

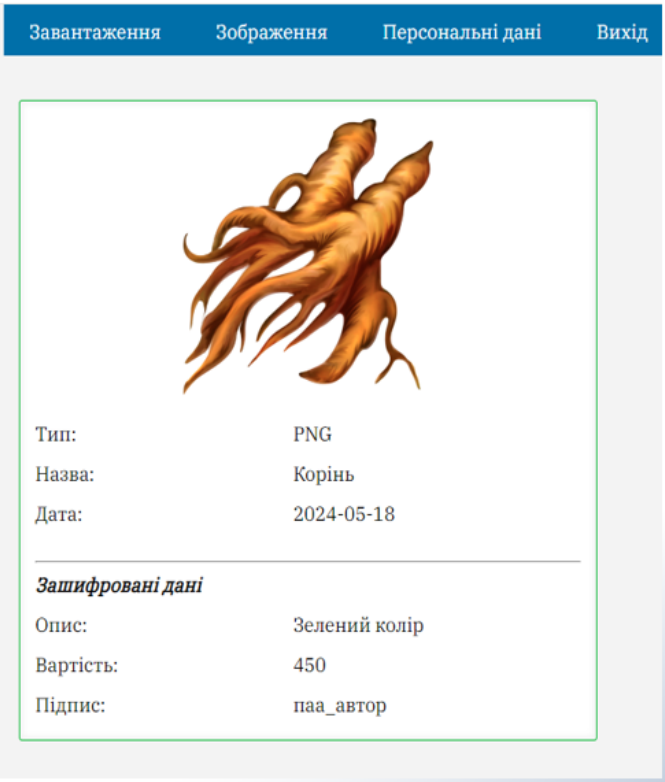
СТРУКТУРА ІНТЕРФЕЙСУ



Файли користувача

Форма для роботи PNG форматом

СТРУКТУРА ІНТЕРФЕЙСУ



Перегляд завантаженого зображення

Перш ніж користувач зможе використовувати програму для приховування інформації в контурах зображень, він повинен авторизуватися в своєму акаунті. Це необхідно для збереження налаштувань, що стосуються приховування інформації у зображеннях.

Після успішного входу в аккаунт користувач має можливість вибрати опцію завантаження растрового зображення для обробки. Вибравши зображення, користувач може завантажити файл для приховування інформації. Це виконується шляхом вибору файлу на комп'ютері та його завантаження на сервер.

13

СТРУКТУРА ІНТЕРФЕЙСУ

```
<form class="c-form" method="POST" action="/user/file" enctype="multipart/form-data"><input name="type" type="text" value="2" class="hide"><span class="label">Дата</span><input name="date" type="date" >
<span class="label">Назва</span><input name="name" type="text" >
<span class="label">Текст</span><input name="text" type="text" >
<span class="label">Вартість</span><input name="sum" type="number" max="10000" min="1"><span class="label">Завантажити зображення</span>
<input type="hidden" name="MAX_FILE_SIZE" value="512000000" />
<input name="load" type="file" required><input class="input-btn" type="submit" value="Приховати дані"></form>
```

Для завантаження нового зображення використовується форма веб-сторінки. У формі знаходяться інпути які приймають перелік параметрів для створення запису і додавання інформації до зображення.

14

ТЕСТУВАННЯ

Ємність приховування визначається кількістю інформації, яку можна приховати в зображенні без значної втрати якості. Це можна оцінити, перевіряючи, скільки біти можна приховати в кожному пікселі.

ВИСНОВКИ

Було успішно розроблено та реалізовано програму приховування інформації у контурах об'єктів растрових зображень, яка забезпечує високий рівень безпеки даних. За допомогою спеціальних алгоритмів приховування інформації, було досягнуто захищеного передавання та зберігання конфіденційної інформації, що зменшує ризик незаконного доступу чи витоку.

Крім того, кожне зображення, що обробляється програмою, забезпечене водяним знаком, що дозволяє відстежувати та ідентифікувати автора чи правовласника цих зображень. Це не лише підвищує безпеку продукту, а й додає додатковий шар захисту від піратства та незаконного використання.

Загалом, розроблена програма стала результатом вдалих досліджень та творчого підходу до розв'язання проблеми зберігання та передачі даних в онлайн-середовищі. Її застосування в реальних умовах може значно поліпшити безпеку та захист інтелектуальної власності в інтернеті.

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ
ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програми приховування інформації у контурах об'єктів растрових зображень

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ: Кафедра менеджменту та безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 96% Схожість 4%

Аналіз звіту подібності (відмітити потрібне):
1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Коваль Н.П.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Бондарчук А.О.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Адлер О.О.
(підпис) (прізвище, ініціали)