Chapter-1

INTRODUCTION

Steganography is an art of secret or invisible communication. The word steganography is derived from the Greek word 'steganos' mean- reticent (hidden) and 'graphein' means "to write".
Information can be hidden in various forms of files, it can be Text, video, audio, or image; these are used as the cover files where data is not visible. The most popular type of steganography is hiding information in a file containing a digital image or picture.
Steganography is not actually a form of cryptography but another way of hiding/protecting the information in transit. Cryptography makes the data unreadable and focuses on keeping content secret whereas steganography keeps the presence of data secret. The advantage of steganography over cryptography is that it does not attract attention to find the file to be suspicious as there is merely any change visible to cover the file.
The method used here for image steganography is LSB(least significant bit) substitution method.

1.1 Image Definition

To a computer, a picture is a collection of numbers that constitute different light intensities in different areas of the image. This numeric representation forms a grid and therefore the individual points are referred to as pixels. Most images on the web consist of a rectangular map of the image's pixels (represented as bits) where each pixel is located and its colour. Pixels are displayed in row x column matrix format.
The number of bits in a colour scheme, called the bit depth, refers to the number of bits used in each pixel. Monochrome and grayscale images use 8 bits for every pixel and are able to display 256 different colours or shades of grey. Digital colour images are typically stored in 24-bit files and use the RGB colour model, also referred to as true colour. All colour variations for the pixels of a 24-bit image are derived from three primary colours: red, green, and blue, and every primary colour is represented by 8 bits. Thus in one given pixel, there are often 256 different quantities of red, green, and blue, adding up to quite 16 million combinations, leading to more than 16 million colours. The file size is directly proportional to the number of colours, more the colours then more will be file size.

1.2 OPENCV

OpenCV is across-platform library where we can use the library and develop real- time computer vision operations. It largely focuses on image processing, video capture and analysis including the features like object detection and face detection.

1.3 NumPy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (example matrices and masked arrays).

1.4 Types

The types module contains the type objects for all objects types defined by the standard interpreter. All the objects of the same type share a single type object.

type()-is a built-in function that returns the type of the objects/data elements stored in any data type or returns a new type object depending on the arguments passed to the function.

Summary: This chapter gives a brief introduction to the project and the libraries called as OpenCV , NumPy, types.

Chapter-2

LITERATURE REVIEW AND OBJECTIVES

2.1 Converting Text To Binary

Information entered by the user as an input to be hidden in the image is first converted into binary. First data type of the text is determined using type() function and then converted to binary format. Example-

"this is text"

01110100 01101000 01101001 01110011 00100000 01101001 01110011 00100000 01110100 01100101 01111000 01110100

2.2 Shifting Image

Translation of an image Rectilinear shift of an image which moves image from one location to another location using shift value of x-axis and y-axis that is number of pixels to shift.

We use cv2.wrapAffine() function to implement the translation.

translation _image=cv2.warpAffine(source, transformation matrix, borderValue)

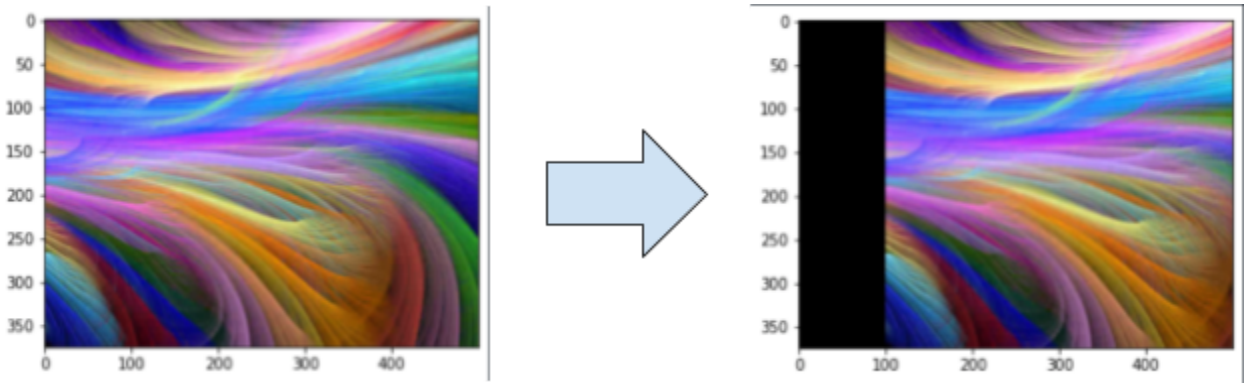

Figure 2.1 shifting pixels

2.3 Filling Blank Space

This translation_image is new shited_image_1: as the whole image is shifted there will be no pixels meaning that there is blank space over there, we will be using that empty space in the image to hide the data, but that empty space is clearly visible in image and that can be suspicious. So we will be filling the empty space with the first appearing pixels in the column of the same row, and that is our shifted_image_2. Now this image works as camouflage and it's difficult to find that this is a steganographic image.

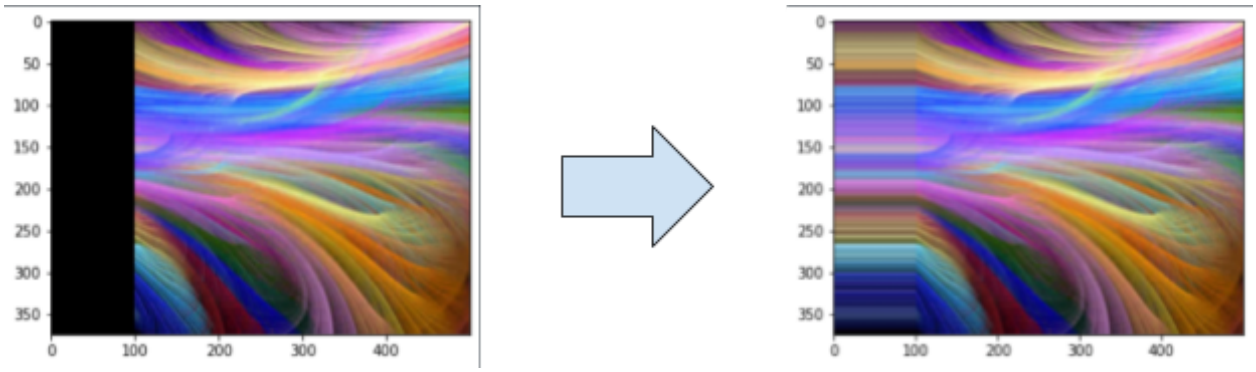

Figure 2.2 Filling blank space with first appearing pixels

2.4 Encoding Information Using Least Significant Bit Method

The least significant bit (LSB) insertion is a common, simple approach to embedding information in a cover image. the least significant bit (8^{th} bit) of the bytes inside a picture is changed to a bit of the secret message. When we use a 24-bit image, a bit of red, green, and blue colour components is often used since they are each represented by a byte. An 800 × 600-pixel image can thus store a complete amount of 1,440,000 bits or 180,000 bytes of embedded data. for instance, a grid for 3 pixels of a 24-bit image is often as follows:

(00101101 00011100 11011100)

(10100110 11000100 00001100)

(11010010 10101101 01100011)

When the amount 200, which binary representation is 11001000, is embedded into the least significant bits of this part of the image, the resulting grid is as follows:

(00101101 00011101 11011100)

(10100110 11000101 00001100)

(11010010 10101100 01100011)

Although the amount was embedded into the first 8 bytes of the grid, only the three underlined bits needed to be changed according to the embedded message. Since there are 256 possible intensities of every primary colour, changing the LSB of the pixel leads to small changes in the intensity of the colours. These changes can't be perceived by the human eye - thus the message is successfully hidden. With a well-chosen image, one can even hide the message within the least as well as second to least significant bit and still not see the difference.

This method is used to encode the binary information computed earlier into the shifted space of the image which has pixels filled from the original image.

2.5 Decoding Steganographic Image

Information encoded in the image can be decoded back from the steganographic image, by extracting the least significant bit from the camouflaged part of the image and converting it back to the text form the same as it was encoded in the image.

Summary: This chapter talks about the different methodology and operations used in the implementation of this project.

Chapter-3

SOFTWARE REQUIREMENT SPECIFICATION

3.1 Overview

The SRS is a specification for a specific software product, program, or set of operations that perform particular functions in a specific environment.SRS is a formal report, which acts as a representation of software that enables the guests to review whether it(SRS) is according to their requirements .

3.2 Software Requirement

- Any IDE to run the program
- A system with Python installed
- Python opencv library
- Python Numpy library
- Python types library

3.3 Hardware Requirement

- Laptop or PC
- I3 processor system or higher
- 4 GB RAM or higher
- 100 GB ROM or higher

Summary: This chapter talks about the software and hardware requirements for successful implementation of this project.

Chapter-4

SYSTEM DESIGN

System design is the process of designing the architecture, components, and interfaces for a system so that it meets the end-user requirements. Quality system design is essential in developing functional and lasting applications.

4.1 Architecture Design

A system's architecture outlines its main parts, their connections (structures), and how they work together. There are several contributing variables to software architecture and design, including business strategy, quality attributes, human dynamics, design, and IT environment.

Figure 4.1 below shows the project's architectural design.

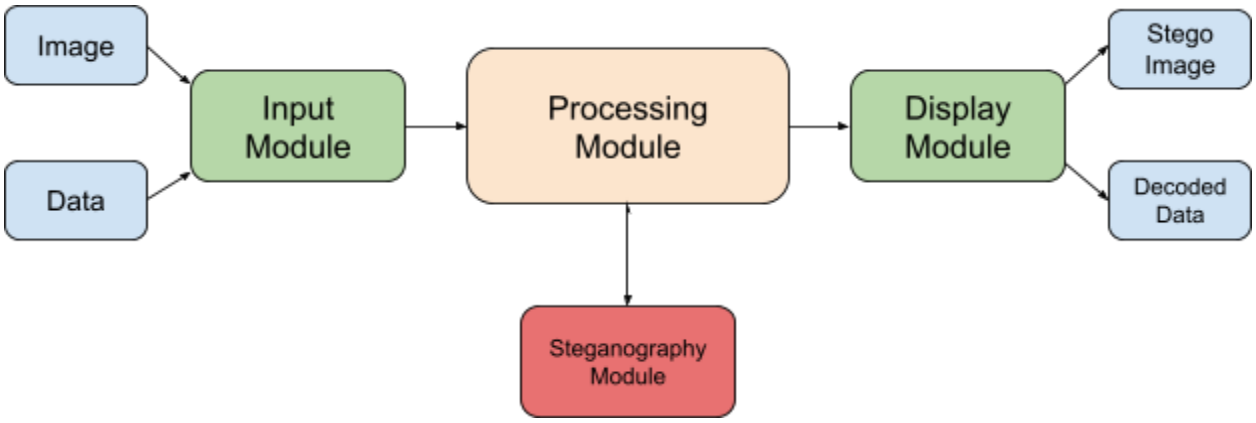

Figure 4.1 Architectural Design

1. **Input Module**: User will be able to give input as an Image and Text.
2. **Processing Module:** This module processes the data given by the user, performs the required operations with the help of a steganography module and passes it to the display module.
3. **Steganography Module:** This module encodes the given text in the image and also decodes the text from the steganographic image.

4. **Display Module:** User will be able to see the outputs of the operations through this module in the form of Image and text.

4.2 Flowchart

An algorithm is graphically represented by a flowchart. It is frequently used by programmers as a technique for planning programs to address issues. It uses interconnected symbols to represent the movement of information and processing. "Flowcharting" is the process of creating a flowchart for a program.

4.2.1 Encoding flowchart

Figure 4.2 shows the project application's encoding flowchart

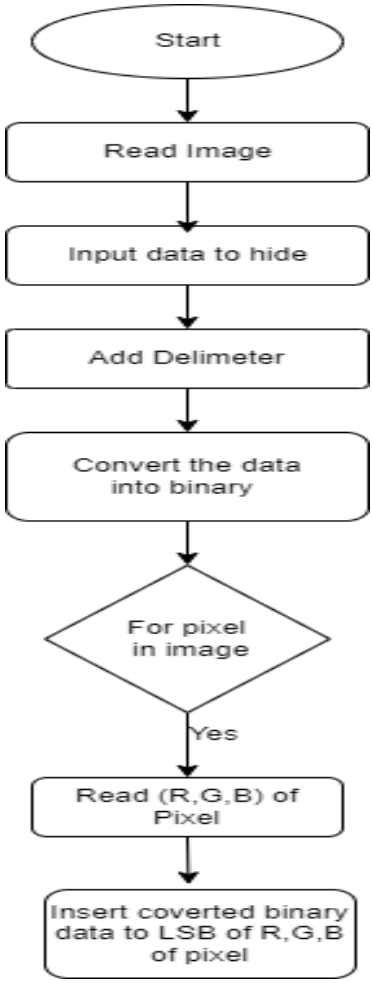

Figure 4.2 Encoding Flowchart

4.2.2 Decoding flowchart

Figure 4.3 shows the project application's decoding flowchart.

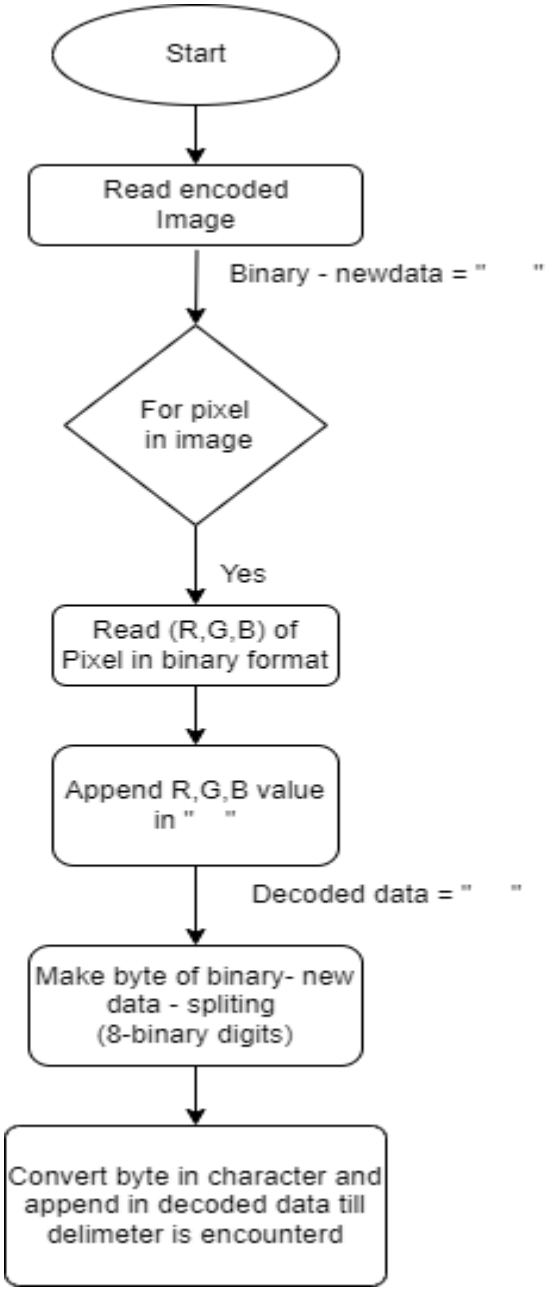

Figure 4.3 Decoding Flowchart

4.3 Use Case Diagram

The scope and high-level functions of a system are described in a use-case diagram. The interaction between the system and its user are also depicted in the diagrams. Use-case diagrams show what the system does and how the user uses it, but they do not show how the system works within.

Figure 4.4 shows the use case diagram of the project.

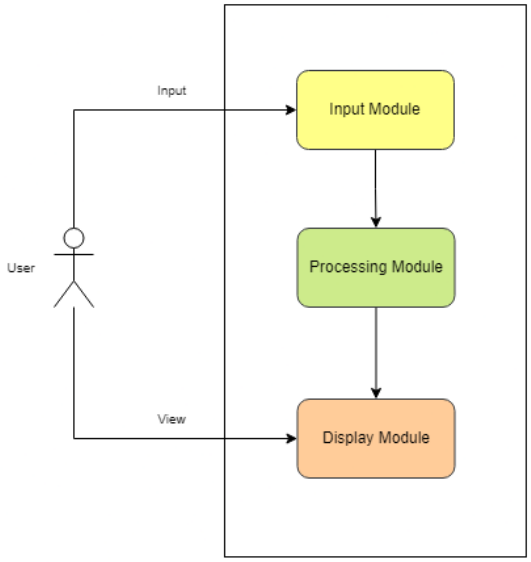

Figure 4.4 Use Case Diagram

1. **Input Module:** User will be able to give input , by passing the image and text to hide.
2. **Processing Module:** This module processes the data given by the user, performs the required operations with the help of a steganography module and passes it to the display module.
3. **Display Module:** User will be able to see the outputs of the operations through this module in the form of Image and text.

Chapter-5

IMPLEMENTATION

5.1 Overview

The processes needed to implement software applications and tools from planning and development to the production stage. It is the process of conceiving, specifying, designing, programming, establishing, testing, and bug fixing involved in creating and maintaining operations, fabrics, or other software factors.

5.2 Algorithm

5.2.1 Encoding Algorithm

Algorithm : Encode
Input - image, data(information to encode)
Output - encoded steganographic image

```
image ← read(input image pixels)
for row in image
    for pixel in row
        R, G, B =  binary_of(pixel)
        while len(binary_data) do:
                pixel[0]=binary_data[i]
                i← i +1
                pixel[1]=binary_data[i]
                i← i +1
                pixel[2]=binary_data[i]
                i← i +1
```

Obtained image is a stego image that contains hidden data/information.

return image

5.2.2 Decoding Algorithm

Algorithm - Decode
Input - steganographic image
Output - decoded message

```
Binary_data ← " "
image ← Read(stego image pixels)
for row in image
    for pixel in row
            R, G, B = binary (pixel)
             Binary_data + =  R[-1]
             Binary_data + =  G[-1]
             Binary_data + =  B[-1]

collect bits from Binary_data  and  convert each 8 bit into character
and store in decoded_data

return decoded_data
```

5.3 Important Code Snippet

5.3.1 Shifting Image

```
def shift_image(img1):
    rows,cols,c=img1.shape
    trastation_matrix=np.float32([[1,0,sx],[0,1,0]])
    img_tr=cv2.warpAffine(img1,trastation_matrix,(cols,rows))
    cv2.imwrite('shifted_image_1.jpg',img_tr)
    edge=[]
    for i in range(rows):
        edge.append(img_tr[i][sx])
    for i in range (rows):
        for j in range(sx):
            img_tr[i][j]= edge[i]
    cv2.imwrite('shifted_image_2.jpg',img_tr)
    return img_tr
```

5.3.2 Converting to binary

```
def to_bin(msg):
    if type(msg) == str:
        return ''.join([format(ord(i), "08b") for i in msg])
    elif type(msg) == bytes or type(msg) == np.ndarray:
        return [format(i, "08b") for i in msg]
    elif type(msg) == int or type(msg) == np.uint8:
        return format(msg, "08b")
    else:
        raise TypeError("Input type not supported")
```

5.3.3 Encode data

```
def hide_data(img, secret_msg):
      nBytes = img.shape[0] * sx * 3 // 8
    print("Maximum Bytes for encoding:", nBytes)
       if len(secret_msg) > nBytes:
           raise ValueError("Error encountered insufficient bytes, need bigger
image or less data!!")
       secret_msg += '$$$$$'
    dataIndex = 0
    bin_secret_msg = to_bin(secret_msg)
    datalen = len(bin_secret_msg)
    for ROW in img:
        i=0
        for pixel in ROW:
            if i<sx:
                r, g, b = to_bin(pixel)
                if dataIndex < datalen:
                          pixel[0] = int(r[:-1] + bin_secret_msg[dataIndex], 2)
                    dataIndex += 1
                if dataIndex < datalen:
                    # LSB of Green pixel
                    pixel[1] = int(g[:-1] + bin_secret_msg[dataIndex], 2)
                    dataIndex += 1
                if dataIndex < datalen:
                    # LSB of Blue pixel
                    pixel[2] = int(b[:-1] + bin_secret_msg[dataIndex], 2)
                    dataIndex += 1
                if dataIndex >= datalen:
                    break
                i=i+1
    return img
```

5.3.4 Decode the image

```
def get_data(img):
    bin_data = ""
    for ROW in img:
        i=0
        for pixel in ROW:
            if i<sx:
                r, g, b = to_bin(pixel)
                bin_data += r[-1]

                bin_data += g[-1]

                bin_data += b[-1]
                i=i+1
    all_bytes = [bin_data[i: i + 8] for i in range(0, len(bin_data), 8)]
    decoded_Data = ""
    for bytes in all_bytes:
        decoded_Data += chr(int(bytes, 2))
        if decoded_Data[-5:] == '$$$$$':
            break

    return decoded_Data[:-5]
```

Chapter-6

RESULTS AND SNAPSHOTS

Snapshot 1: The following figure 6.1 shows change in file size after encoding

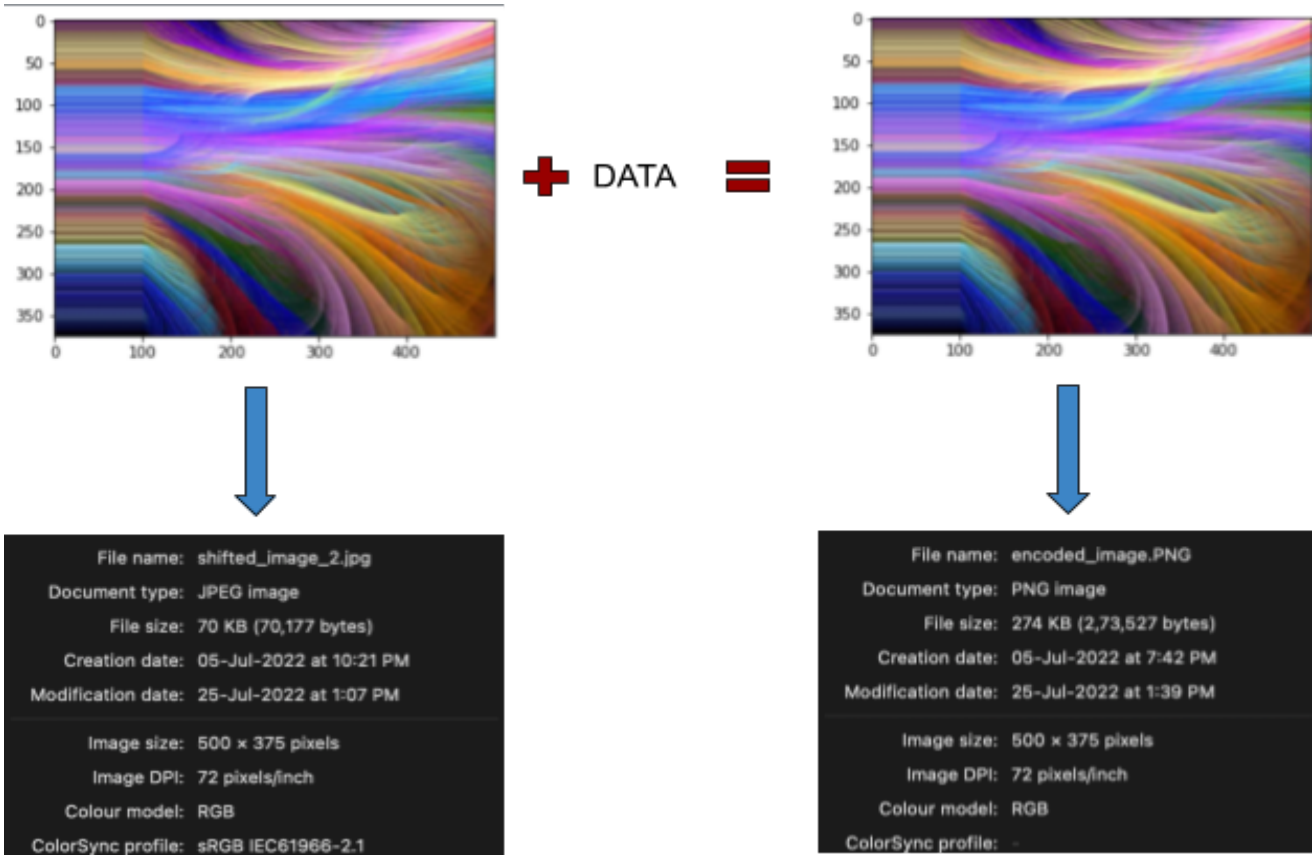

Figure 6.1 Comparing file size after encoding

Snapshot 2 : Interactive option for Encoding

```
Image Steganography
1. Encode the data
2. Decode the data
Select the option: 1

Encoding...
Enter shift value :100
Enter image name (with extension): original.jpg
Enter data to be encoded: This is the data to be encoded
Enter the name of the new encoded image (with extension): encoded_image.PNG
Maximum Bytes for encoding: 96000
```

Figure 6.2 Data encoding

Snapshot 2 : Interactive option for Decoding

```
Image Steganography
1. Encode the data
2. Decode the data
Select the option: 2

Decoding...
Enter shift value :100
Enter the name of the Steganographic image (with extension): encoded_image.PNG
Decoded message is  This is the data to be encoded
```

Figure 6.3 Image decoding

Chapter-7

CONCLUSION

Steganography is the method of hiding information in such a way that its presence is not noticed. This report discusses the LSB method to hide the secret message in the Least Significant bit of the image. The LSB modification technique gives an easy way to embed information in images, but the data can be easily decoded. LSB method is applied for various file formats. This method can be used for both JPG and PNG file formats.

REFERENCE

1. J. N. Cheltha C, M. Rakhra, R. Kumar and H. Walia, "A Review on Data hiding using Steganography and Cryptography," 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2021, pp. 1-4, DOI: 10.1109/ICRITO51393.2021.9596531.
2. S. M. Masud Karim, M. S. Rahman and M. I. Hossain, "A new approach for LSB based image steganography using the secret key," 14th International Conference on Computer and Information Technology (ICCIT 2011), 2011, pp. 286-291, DOI: 10.1109/ICCITechn.2011.6164800.
3. K. Thangadurai and G. Sudha Devi, "An analysis of LSB based image steganography techniques," 2014 International Conference on Computer Communication and Informatics, 2014, pp. 1-4, doi: 10.1109/ICCCI.2014.6921751.
4. https://numpy.org/doc/stable/user/whatisnumpy.html
5. https://www.oreilly.com/library/view/python-standard-library/0596000960/ch01s16.html
6. https://www.javatpoint.com/software-requirement-specifications
7. https://stackoverflow.com
8. https://pythonguides.com/numpy
9. https://www.toppr.com/guides/python-guide/references/methods-and-functions/methods/built-in/type/python-type/