Dharmsinh Desai University, Nadiad

Faculty of Technology

Department of Computer Engineering

B. Tech – CE, Semester: VI

Project title: Image Steganography

Subject: System Design Practice

Prepared by: Birva Babaria (CE010, 19CEUON064)
Khushi Doshi (CE038, 19CEUEG038)

Guided by: Prof. Shaifali P. Malukani

CERTIFICATE

This is to certify that the project entitled as **"Image Steganography"** carried out in the subject of **System Design Practice** and recorded in this report is bonafide work of

Birva Babaria (Roll No.: CE010, ID: 19CEUON064)

Khushi Doshi (Roll No.: CE038, ID: 19CEUEG038)

Of **B.Tech Semester 6th** in the branch of Computer Engineering, under the guidance and supervision of **Prof. Shaifali P. Malukani** during the academic year 2021-2022.

Prof. Shaifali P. Malukani
(Project Guide and Assistant Professor)
Faculty of Technology,
Dharmsinh Desai University,
Nadiad.

Dr. C.K. Bhensdadia
Head of CE Dept.,
Faculty of Technology,
Dharmsinh Desai University,
Nadiad.

Contents

Abstract

The objective of steganography is to send a message through some innocuous carrier to a receiver while preventing anyone else from knowing that message. Many different carrier file format can be used, but digital images are most popular because of their frequency on the internet. Different users have different requirements. For example some user may require absolute invisibility of the secret message while others may require larger secret message to be hidden.

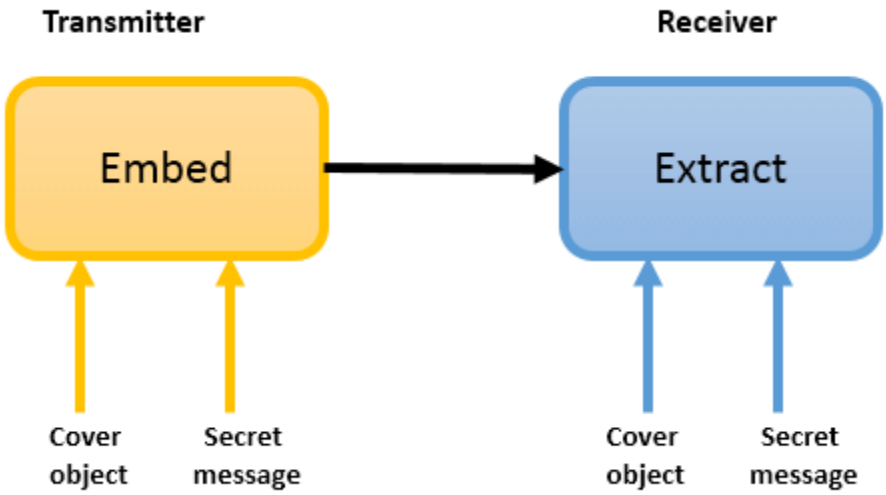
This project hides text/image with the image. For more convenient approach, all the channels (Red, Green and Blue) of few last consecutive bits are chosen rather than single least significant bit (LSB). Sender can select the cover image and secret text/image file to generate a stego image. Receiver can further use that stego image to decode the secret text/image. Stego image cannot be decoded by any random decrypting algorithm.

Chapter 1

INTRODUCTION

1.1) What is Steganography?

"Steganography is the art and science of embedding secret messages in cover message in such a way that no one, apart from sender and intended recipient, suspects the existence of the message." The word steganography is derived from the Greek word steganos (meaning hidden or covered) and the Greek root graph (meaning to write). Thus, steganography is not only the art of hiding data but also hiding the fact of transmission of secret data. Steganography is also defined as the study of the invisible communication. Steganography usually deals with the ways of hiding the existence of the communicated data in such a way that it remains confidential.



(Figure 1.1) Process of steganography

Figure 1.1 depicts the basic algorithm of how data is embedded and extracted by transmitter and receiver respectively.

In Image steganography secrecy is achieved by embedding data into the cover image which results in generation of stego image. There are different types of steganography techniques each of them have their strength and weaknesses. In this project, we study different types of steganography techniques where different types of secret data can be handled as per requirement. Depending on the nature of the cover image, steganography can be divided into five types:

- Text Steganography
- Image Steganography
- Video Steganography
- Audio Steganography
- Network Steganography

In today's world, the communication is basic necessity of every growing area. Everyone wants the secrecy and safety of their communicating data. In our daily life, we use many secure pathways like internet or telephone for transferring and sharing information, but it is not safe at certain level. In order to share the information in a concealed manner two techniques could be used. These mechanisms are 'cryptography' and 'steganography'. In cryptography the message is encrypted using the encryption key which is known to sender and receiver only. The message cannot be accessed by anyone else without using the encryption key. In steganography the process of hiding data into any multimedia content like image, audio, video referred as a embedding. For increasing confidentiality of the message of the techniques can be combined.

1.2) History

In ancient Greece, wax tablets with a wooden backing were common writing surfaces since the wax could be melted and reused. To send the secret messages, a message would be inscribed directly on the wooden tablet before the wax was applied. Another, innocuous message would be carved into the wax on the top and the tablet is sent to its intended recipient, who would melt off the wax and read the true message.

Another method of concealing messages was to encode a message on the piece of thread using Morse code. This thread would be knitted into the clothing of a messenger and then removed and read at its destination.

During the World Wars, many different methods of sending hidden messages were used. Female spies would encode messages in knitted patterns (leading to a ban on new knitting patterns). Photosensitive glass (which shows an image when exposed to the correct wavelength of light) was used during World War II to send messages to Allied forces.

During World War II the Germans introduced microdots, which were complete documents, pictures, and plans reduced in size to the size of a dot and were attached to normal paperwork.

1.3) Applications

Various application of steganography techniques are as follow:

1) Confidential communication and secret data storing
2) Protection of data alteration
3) Access control system for digital content distribution
4) Media database systems

Chapter 2

REQUIREMENT ANALYSIS

2.1) Functional requirements

Functional requirements are the requirements that define specific behaviour or function of the system.

- Secret Text Message: Sender has to provide the message which is supposed to be hidden in the cover image. Sender can provide message by speaking in microphone or by manually typing it.
- Secret Hidden Image: Sender has to provide the secret image by browsing in machine's directory.
- Cover Image: Cover image is selected by the sender in which text/image is supposed to be hidden.
- Stego Image: Receiver has to provide appropriate stego image for decoding text/image from the cover image.
- LSB algorithm implementation of encryption and decryption to hide the text in image and image in image.

2.2) Non-Functional requirements

- Safety requirements: Sender and receiver should ensure that they are using same software for encrypting and decrypting data inside image. Both should take care of eavesdropping.
- Security requirements: This software is embedding secret data in cover image. Only sender and receiver should be aware of encrypted file. User should not unfold the message regarding sent image as well as receiver information.
- Software Quality Attributes: The quality of software is maintained in such a way that only sender and receiver can communicate through the image. There is no probability of some other person to know about this communication.

2.3) Software requirements specifications

1) Hide Text In Image

R.1.1: Home

Description: This function takes user to the home page of 'hide text in image' module.

Input: Click on the 'Hide text in image' button.

Output: Home page of ‘hide text in image’ module would be displayed where user would be able to encode or decode the text as per his choice.

R.1.2: Encode text

Description: This function allows user to encode the provided text in the cover image.

Input: Enter text manually or using microphone, provide the cover image and click on encode button.

Output: Text would be encoded within the cover image and result would be displayed.

R.1.2.1: View Image

Description: This function displays the stego image on full screen.

Input: Click on the view image button.

Output: Full screen stego image would be displayed.

R.1.2.2: View PSNR value

Description: This function will show the calculated PSNR value of the stego image.

Input: Click on the PSNR value button.

Output: Alert box would be appeared with PSNR value.

R.1.2.3: Download Image

Description: This function will allow user to download the stego image.

Input: Click on the Download button.

Output: Stego image would be downloaded on user’s machine.

R.1.2.4: Back to home page

Description: This function takes user back to the home page of ‘hide text in image’ module.

Input: Click on the back button.

Output: Home page of ‘hide text in image’ module would be displayed.

R.1.3: Decode text

Description: This function allows user to decode the secret text from the provided stego image.

Input: Browse the stego image from the machine and click on the decode button.

Output: Text would be decoded and appropriate results would be displayed.

R.1.3.1: Decoded text

Description: The text decoded in the stego image is displayed on the screen. If there was no hidden text in the stego image, 'No hidden text found' would be displayed.

R.1.3.2: Back to home page

Description: This function takes user back to the home page of 'hide text in image' module.

Input: Click on the back button.

Output: Home page of 'hide text in image' module would be displayed.

R.1.4: Back to main page

Description: This function takes user back to the home page where he can choose among both modules

Input: Click on the back button.

Output: Home page with different modules button would be displayed.

2) Hide Image In Image

R.2.1: Home

Description: This function takes user to the home page of 'hide image in image' module.

Input: Click on the 'Hide image in image' button.

Output: Home page of 'hide image in image' module would be displayed where user would be able to encode or decode the image as per his choice.

R.2.2: Encode image

Description: This function allows user to encode the provided secret image in the cover image.

Input: Browse secret image and cover image from the machine and click on encode button.

Output: Secret image would be encoded within the cover image and result would be displayed.

R.2.2.1: View Image

Description: This function displays the stego image on full screen.

Input: Click on the view image button.

Output: Full screen stego image would be displayed.

R.2.2.2: View PSNR value

Description: This function will show the calculated PSNR value of the stego image.

Input: Click on the PSNR value button.

Output: Alert box would be appeared with PSNR value.

R.2.2.3: Download Image

Description: This function will allow user to download the stego image.

Input: Click on the Download button.

Output: Stego image would be downloaded on user's machine.

R.2.2.4: Back to home page

Description: This function takes user back to the home page of 'hide image in image' module.

Input: Click on the back button.

Output: Home page of 'hide image in image' module would be displayed.

R.2.3: Decode Image

Description: This function allows user to decode the secret image from the provided stego image.

Input: Browse the stego image from the machine and click on the decode button.

Output: Secret image would be decoded and appropriate results would be displayed.

R.2.3.1: View Image

Description: This function displays the secret image on full screen.

Input: Click on the view image button.

Output: Full screen secret image would be displayed.

R.2.3.2: Download Image

Description: This function will allow user to download the secret image.

Input: Click on the Download button.

Output: Secret image would be downloaded on user's machine.

R.2.3.3: Back to home page

Description: This function takes user back to the home page of 'hide image in image' module.

Input: Click on the back button.

Output: Home page of 'hide image in image' module would be displayed.

R.2.4: Back to main page

Description: This function takes user back to the home page where he can choose among both modules

Input: Click on the back button.

Output: Home page with different modules button would be displayed.

2.4) Technologies/Tools used

Technologies:

- Django (python framework in which this application is build)
- Python (language used to implements several functionalities)
- Html (to write the content that would be displayed n web page)
- Jinja (to access data sent from the webpage in html page)
- CSS (used to design the web page)
- Bootstrap (used to design the web page)

Tools:

- Git (used to upload the project work in git repository)
- Visual studio code (platform from which the project is implemented)

Chapter 3

IMAGE STEGANOGRAPHY

3.1) Types of steganography

Depending on the nature of the cover object (actual object in which secret data is embedded) steganography can be divided into five types:

3.1.1) Text steganography

Text steganography is hiding the information inside the text files. In this method, secret data is hidden behind every nth letter of every word of text message. It also involves things like changing the format of the text, changing words within the text, generating random character sequences or using context-free grammars to generate readable texts. There are various techniques to hide the data in the text files. Some of them are:

- Format Based Method
- Random and statistical Generation
- Linguistic Method

3.1.2) Image steganography

Image steganography is hiding the data taking the cover object as the image. In image steganography pixel intensities are used to hide the data. In digital steganography, images are widely used cover source because there are a huge number of bits present in the digital representation of an image. There are various techniques to hide the data in the image files. Some of them are:

- Least Significant Bit Insertion (LSB)
- Masking and Filtering
- Redundant Pattern Encoding
- Encrypt and scatter
- Coding and Cosine Transformation

3.1.3) Audio steganography

In audio steganography, the secret message is embedded into an audio signal which alters the binary sequence of the corresponding audio file. Hiding secret messages in the digital sound is a much more difficult process when compared to others, such as Image Steganography. There are various techniques to hide the data into the audio files. These methods hides the data in WAV, AU, and even MP3 sound files. Some of them are:

- Least Significant Bit Encoding
- Parity Encoding
- Phase Coding
- Spread Spectrum

3.1.4) Video steganography

In video steganography you can hide data into digital video format. The advantage of this type is large amount of data can be hidden inside and the fact that it is a moving stream of images and sounds. You can think of this as the combination f Image Steganography and Audio Steganography. Two man classes of video steganography include:

- Embedding data in uncompressed raw video and compressing it later
- Embedding data directly into the compressed data stream

3.1.5) Network steganography

In network steganography embedding information within network control protocols used in data transmission such as TCP, UDP, ICMP etc. you can use steganography in some covert channels (not openly displayed or acknowledged) that you can find in the OSI model. For example, you can hide information in the header of a TCP/IP packet in some field that are either optional.

3.2) Steganography in image

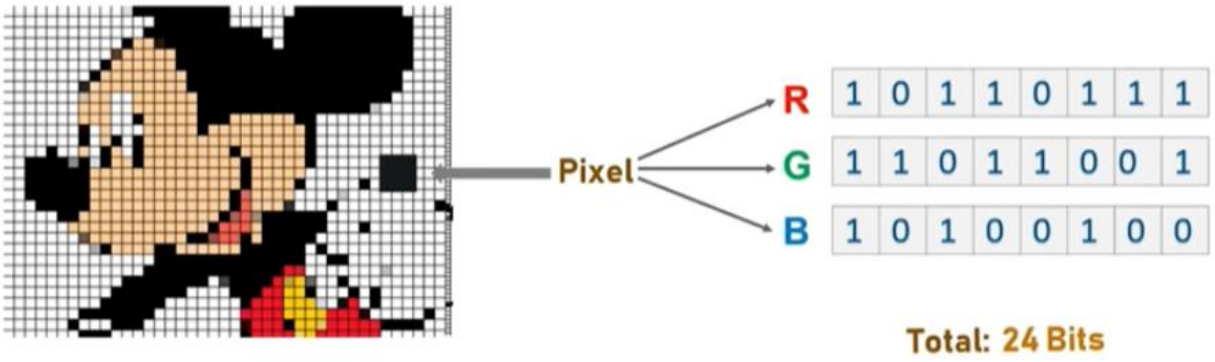
As the name suggests, Image Steganography refers to the process of hiding data within the image file. The image selected for this purpose is called the cover image and the image obtained after the steganography is called as the stego image. An image with a secret message inside can easily be spread over the World Wide Web (WWW). Image steganography is the technique of hiding the data in such a way that prevents the unintended user from the detection of the hidden messages or data.

The current project aims to use image steganography using 'Least Significant Bit' technique. Text and image files are been used as the secret information. This hidden information can only be retrieved through proper decoding technique. Hence, the user has to make sure information is encoded and decoded with same compatible algorithms. The encryption and decryption of the images is done using the python code.

3.2.1) Concept of LSB based data embedding

When we talk about image steganography, the idea is quiet simple. Images are made up of pixels which usually refer to the color of that particular pixel. We can describe a digital image as a finite set of digital values, called pixels. Pixels are the smallest individual element of an image, holding values that represents the brightness of a given color at any specific point. So we can think of image as a matrix (two-dimensional array) of pixels which contains a fixed number of rows and columns.

LSB stands for Least Significant Bit. The idea behind LSB embedding is that if we change the last bit value of the pixel, there won't be much visible change in the colour. In this technique last pixel bit is modified and replaced with the secret message's data bit.



(Figure 3.1) Representation of a pixel

Figure 3.1 depicts the channels of each pixel in an image. Each pixel is divided into three channels i.e., Red, Green and Blue (RGB).

(Figure 3.2) Comparison of modification in MSB and LSB

Figure 3.2 depicts results of Most Significant Bit or Least Significant Bit modification

From above image it is clear that, if we change MSB it will have a larger impact on the final value but if we change the LSB the impact on the final value is minimal, thus we use least significant bit steganography.

3.2.2) How LSB technique works?

Each pixel in an image contains three values which are Red, Green, Blue, these value ranges from 0-255, in other words, they are 8-bit values. Let's take an example of how this technique works, suppose you want to hide the message in 4x4 image which has the following pixel values:

[(225, 12, 99), (155, 2, 50), (99, 51, 15), (15, 55, 22), (155, 61, 87), (63, 30, 17), (1, 55, 19), (99, 81, 66), (219, 77, 91), (69, 39, 50), (18, 200, 33), (25, 54, 190)]

Using the ASCII Table, we can convert the secret message into decimal values and then into binary values. Let's assume our message is: **0110100 0110101.** Now, we iterate over the pixel values one by one after converting them to binary values. We replace each least significant bit with that message bits sequentially (e.g., 255 is 11100001, we replace the last bit, and the bit in the right (1) with the first data bit (0) and so on). This will only modify the pixel values by +1 or -1 which is not noticeable at all. The resulting pixel values after performing LSB modification are shown as below:

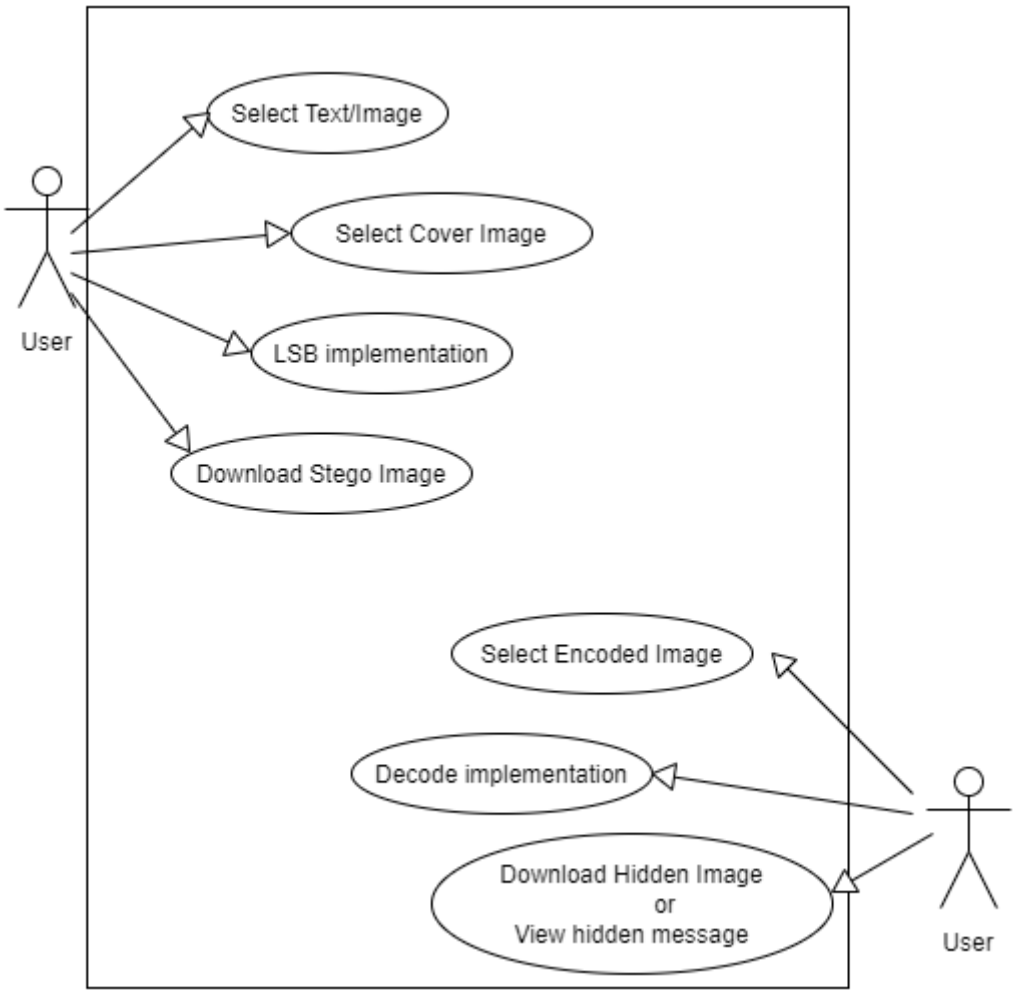
[(224, 13, 99),(154, 3, 50),(98, 50, 15),(15, 54, 23),(154, 61, 87),(63, 30, 17),(1, 55, 19),(99, 81, 66),(219, 77, 91),(69, 39, 50),(18, 200, 33),(25, 54, 190)]

Chapter 4

DESIGN

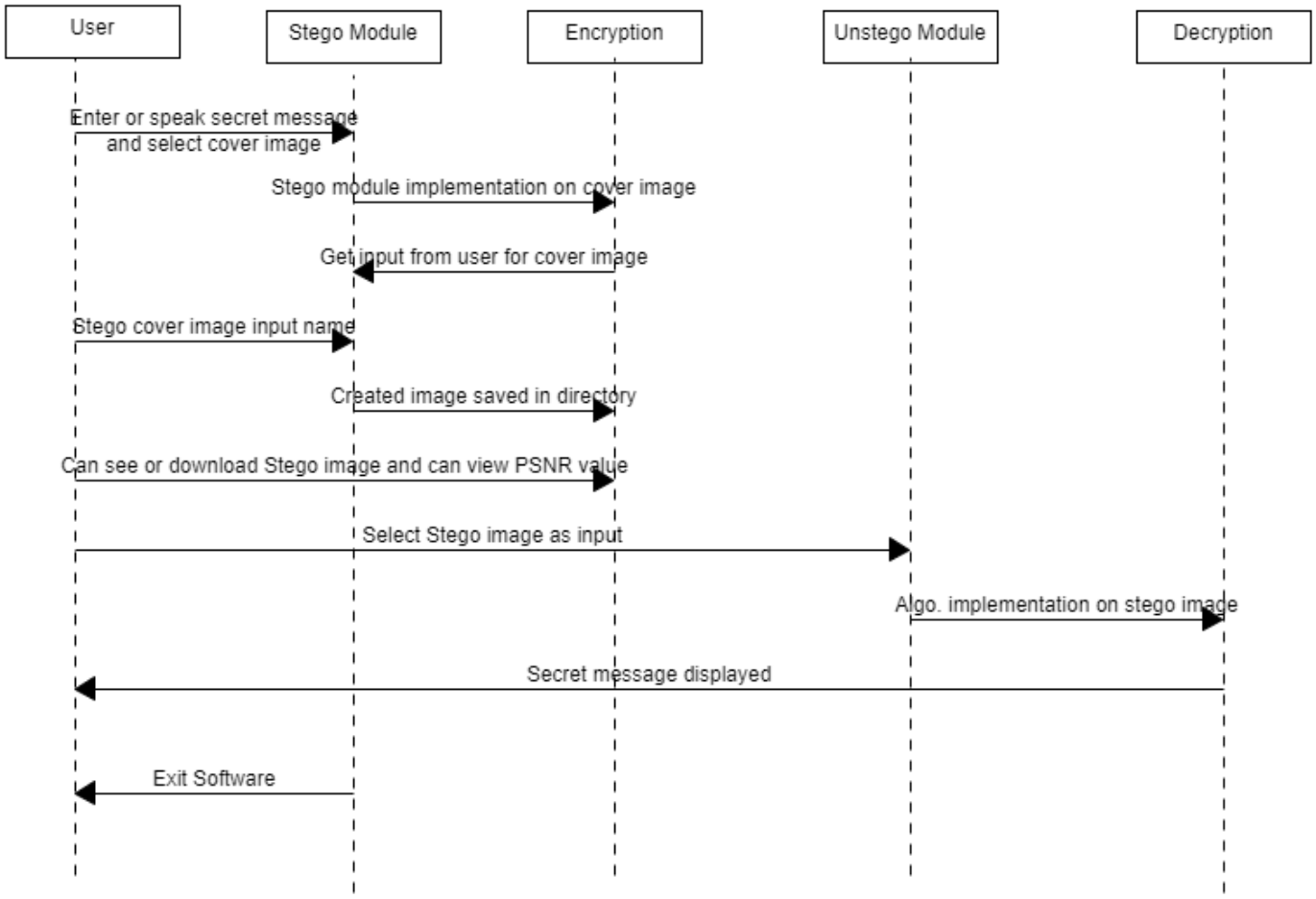
4.1) Use case diagram

In this diagram one actor depicts encoding part of the steganography and another actor depicts the decoding part of the steganography.



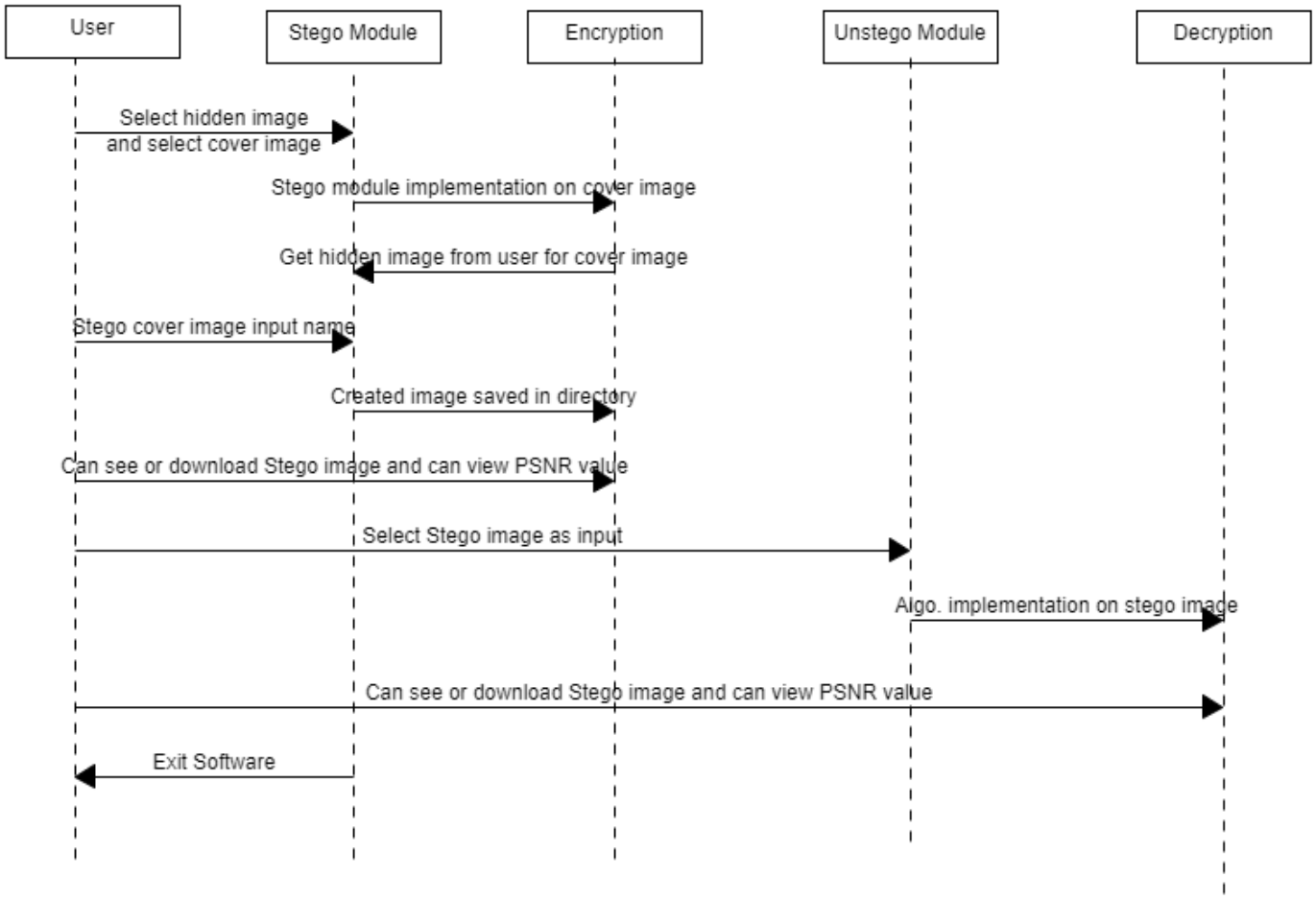
4.2) Sequence diagram (hide text in image)

This diagram conveys the process of steganography implementation of ‘hide text in image’ module.



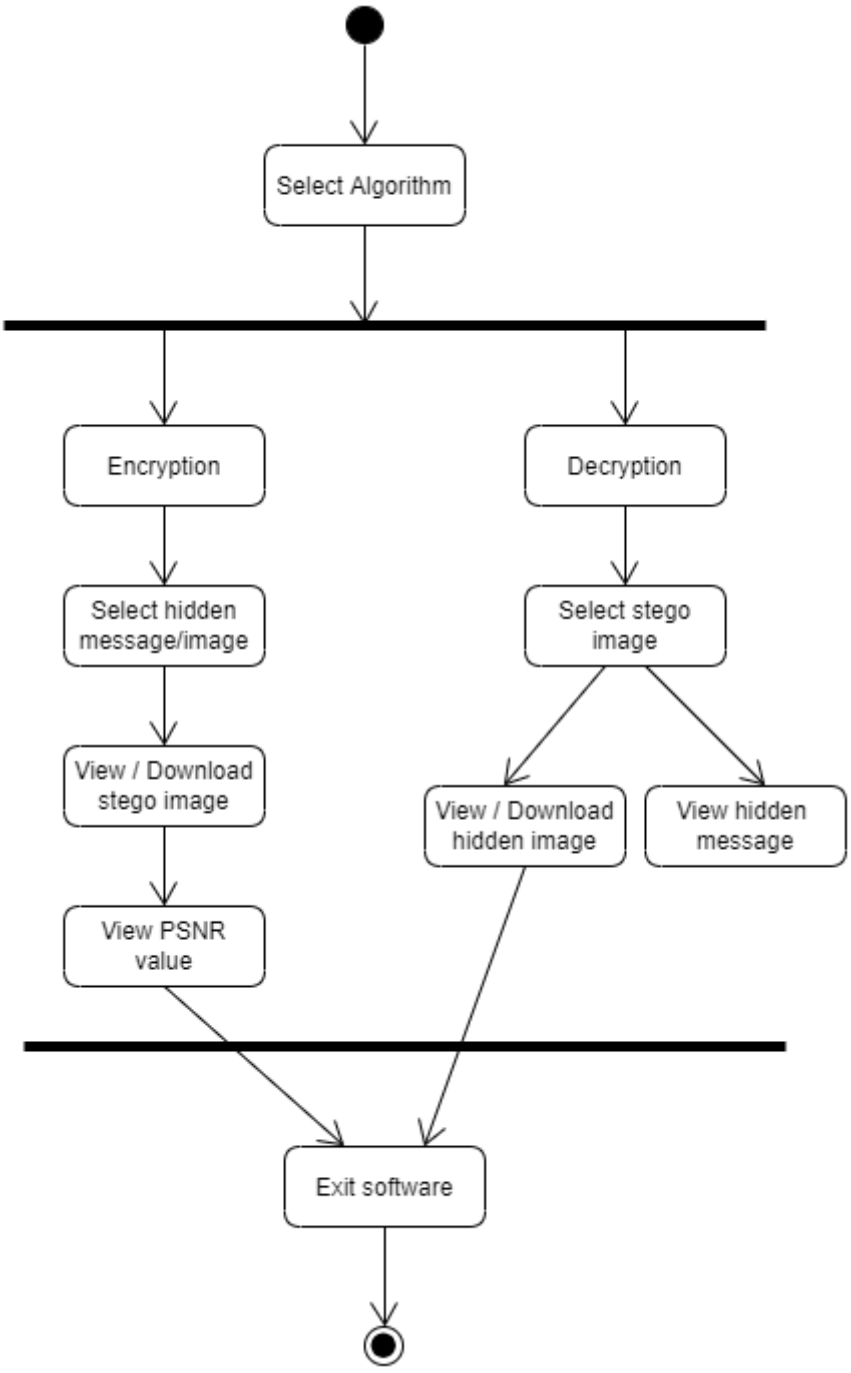
4.3) Sequence diagram (hide image in image)

This diagram conveys the process of steganography implementation of ‘hide image in image’ module.



4.4) Activity diagram

This diagram conveys the flow of encryption and decryption functionalities of both the modules.



Chapter 5

IMPLEMENTATION DETAILS

5.1) Modules

Each module consist of several implemented algorithms that are required to fulfil certain functionalities. The system consists of two basic modules:

- HideTextInImage module
- HideImageInImage module

5.1.1) 'Hide text in image' module

This module uses the LSB implementation to hide the secret text into the cover image. Python libraries like PIL and NumPy are also been used. Before proceeding to any algorithm we allow user to choose whether they would like to perform encoding or decoding.

Encoding: Firstly, we write the code to convert the source image into NumPy array of pixels and store the size of the image. We check if the mode of the image is RGB or RGBA and consequently set n. We also compute total pixels. Further, we add delimiter ("$ka1b2") at the end of the secret message to identify the end of the text. We convert this updated message to binary form and calculate the required pixels. Then we check if total pixels available is sufficient for the secret message or not. If yes, we iterate through each pixel one by one and change the LSB of the pixel with the data of the secret text. Finally, we have the updated pixels array and we can use this to create and save it as the destination output image.

Decoding: Firstly, we repeat a similar procedure of saving the pixels of the source image as an array, figuring out the mode, and calculating the total pixels. Secondly, we need to extract the least significant bit from each of the pixels starting from top-left of the image and store it in groups of 8 bits. Further, we convert these groups into ASCII characters to find the hidden message until we read the delimiter inserted previously completely. Finally, we check if there was a delimiter at the end or not. If not, that means there was no hidden message in the image.

5.1.2) 'Hide image in image' module

This module uses the modified LSB implementation to hide the secret image in the cover image. PIL python library is used along with certain extra user-defined functions. Before proceeding to any algorithm we allow user to choose whether they would like to perform encoding or decoding.

Encoding: Firstly, user provides two images, an image we want to conceal and an image we want to use for concealing. The number of pixels in the image used for hiding an image must be atleast (2 * number of pixels in the image to be hidden + 1). Secret image is converted into a string with

concatenated binary numbers representing the RGB channel values of all pixels in the image, padded with leading zeros when necessary. Hence, each pixel is 24 bits long. Secondly, the last 4 bits of every pixel of cover image is replaced with the pixel string for hidden image. The first pixel in the top left corner is used to store the width and height of the image to be hidden, which is necessary for the recovery of the hidden image. Finally, we have the RGB image which is the copy of cover image with altered 4 least significant bits and we can use this to save it as the destination output image.

Decoding: Firstly, the stego image is loaded to recover a hidden image. Width and the height of the hidden image are determined by the first pixel of the stego image. Further, the information representing a hidden image is stored in the 4 least significant bits of a subset of pixels of the visible image. A binary string is created that represents those pixels. Then the image is reconstructed using the height, width and pixel string. If the image cannot be decoded then appropriate error message is generated. Reconstructed decoded image can be saved at its output destination or can be downloaded by the user.

5.2) Major Functionalities

5.2.1) Functionalities of 'hide text in image'

- **Encode:** This function encodes the secret text into the cover image. Exception is handled accordingly if the length of the text is too large to hide.

```
def encode(request):
    if request.method=="POST":
        myfile=request.FILES['myfile']
        message=request.POST['message']
        BASE_DIR = Path(__file__).resolve().parent.parent
        fs=FileSystemStorage()
        filename=fs.save(myfile.name,myfile)
        url=fs.url(filename)
        url1=str(BASE_DIR)+url
        original = cv2.imread(url1)
        img = Image.open(url1, 'r')
        width, height = img.size
        array = np.array(list(img.getdata()))
        if img.mode == 'RGB':
            n = 3
        elif img.mode == 'RGBA':
            n = 4
        total_pixels = array.size//n
        message += "$ka1b2"
        b_message = ''.join([format(ord(i), "08b") for i in message])
        req_pixels = len(b_message)
        messageError=''
        if req_pixels > total_pixels:
            messageError="ERROR: Need larger Image size"
            request.session['messageError']=messageError
            return
render(request,"homeText.html",{'reqPixel':req_pixels,'totalPixel':total_pixels})
        else:
            index=0
            for p in range(total_pixels):
                for q in range(0, 3):
                    if index < req_pixels:
                        array[p][q] = int(bin(array[p][q])[2:9] + b_message[index], 2)
                        index += 1

            array=array.reshape(height, width, n)
            enc_img = Image.fromarray(array.astype('uint8'), img.mode)
            url2=str(BASE_DIR)+url
            enc_img.save(url2)
            message1="Message Encoded Successfully"
            fname=str(BASE_DIR)+str(MEDIA_URL)+filename

            compressed = cv2.imread(fname)
            mse = np.mean((original - compressed) ** 2)
            if(mse == 0):
```

```
            psnr = 100
        else:
            max_pixel = 255.0
            psnr = 20 * log10(max_pixel / sqrt(mse))
        return
render(request,"encDownload.html",{'url':filename,'message':message,'message1':message1,'PSNR':psnr})
```

- **Decode:** This function decodes the text from the stego image. If no text is found then error message is displayed accordingly.

```
def decode(request):
        if request.method=="POST":
            myfile=request.FILES['myfile']
            BASE_DIR = Path(__file__).resolve().parent.parent
            fs=FileSystemStorage()
            filename=fs.save(myfile.name,myfile)
            url=fs.url(filename)
            url1=str(BASE_DIR)+url
            img = Image.open(url1, 'r')
            array = np.array(list(img.getdata()))
            if img.mode == 'RGB':
                n = 3
            elif img.mode == 'RGBA':
                n = 4
            total_pixels = array.size//n
            hidden_bits = ""
            for p in range(total_pixels):
                for q in range(0, 3):
                    hidden_bits += (bin(array[p][q])[2:][-1])
            hidden_bits = [hidden_bits[i:i+8] for i in range(0, len(hidden_bits), 8)]
            message = ""
            for i in range(len(hidden_bits)):
                if message[-6:] == "$ka1b2":
                    break
                else:
                    message += chr(int(hidden_bits[i], 2))
            msg = ""
            err_msg = ""
            if "$ka1b2" in message:
                msg = message[:-6]
            else:
                err_msg = "No Hidden Message Found"
            return render(request,"decDownload.html",{'msg':msg, 'err_msg': err_msg})
```

- **PSNR value:** This function calculates the PSNR value of the stego image in order to check the distortion in the visible image.

```
def PSNR(original, compressed):
    mse = np.mean((original - compressed) ** 2)
    if(mse == 0):
        return 100
    max_pixel = 255.0
    psnr = 20 * log10(max_pixel / sqrt(mse))
    return psnr
```

5.2.2) Functionalities of 'hide image in image'

- **Encode:** This function encodes the secret image into the cover image. Exception is handled accordingly if the length of the cover image is not as twice as the length of secret image.

```
def encode(img_visible, img_hidden):
    encoded_image = img_visible.load()
    img_hidden_copy = img_hidden.load()
    width_visible, height_visible = img_visible.size
    width_hidden, height_hidden = img_hidden.size
    hidden_image_pixels = get_binary_pixel_values(
        img_hidden_copy, width_hidden, height_hidden)
    encoded_image = change_binary_values(
        encoded_image, hidden_image_pixels, width_visible, height_visible, width_hidden,
height_hidden)
    return img_visible

def encodeImage(request):
    if request.method == "POST":
        imageToHide = request.FILES['imageToHide']
        hideImage = request.FILES['hideImage']
        BASE_DIR = Path(__file__).resolve().parent.parent
        fs = FileSystemStorage()
        filename1 = fs.save(imageToHide.name, imageToHide)
        url_1 = fs.url(filename1)
        url1 = str(BASE_DIR)+url_1
        filename2 = fs.save(hideImage.name, hideImage)
        url_2 = fs.url(filename2)
        url2 = str(BASE_DIR)+url_2
        original=cv2.imread(url2)
        img_visible = Image.open(url2, 'r')

        array = np.array(list(img_visible.getdata()))
        if img_visible.mode == 'RGB':
            n = 3
        elif img_visible.mode == 'RGBA':
            n = 4
        total_pixels = array.size//n
        img_hidden = Image.open(url1, 'r')

        array1 = np.array(list(img_hidden.getdata()))
```

```
        if img_visible.mode == 'RGB':
            n = 3
        elif img_visible.mode == 'RGBA':
            n = 4
        required_pixels = array1.size//n

        if required_pixels >= total_pixels:
            messageError = "ERROR: Need larger image size"
            request.session['messageError1'] = messageError
            return render(request,
"homeImage.html",{'reqPixel':required_pixels,'totalPixel':total_pixels})
        else:
            encoded_image = encode(img_visible, img_hidden)
            encoded_image.save(url2)
            message1 = "Image Encoded Successfully"
            fname=str(BASE_DIR)+str(MEDIA_URL)+filename2
            compressed = cv2.imread(fname)
            mse = np.mean((original - compressed)**2)
            if(mse == 0):
                psnr = 100
            else:
                max_pixel = 255.0
                psnr = 20 * log10(max_pixel / sqrt(mse))
            return render(request, "encImage.html", {'url1': filename1, 'url2': filename2,
'message1': message1,'PSNR':psnr})
```

- **Decode:** This function decodes the secret image from the stego image. If no image is found then error message is displayed accordingly.

```
def decode(image):
    image_copy = image.load()
    width_visible, height_visible = image.size
    r, g, b = image_copy[0, 0]
    r_binary, g_binary, b_binary = rgb_to_binary(r, g, b)
    w_h_binary = r_binary + g_binary + b_binary
    width_hidden = int(w_h_binary[0:12], 2)
    height_hidden = int(w_h_binary[12:24], 2)
    pixel_count = width_hidden * height_hidden
    hidden_image_pixels = extract_hidden_pixels(
        image_copy, width_visible, height_visible, pixel_count)
    decoded_image = reconstruct_image(
        hidden_image_pixels, width_hidden, height_hidden)
    return decoded_image


def decodeImage(request):
    if request.method == "POST":
        try:
            decFile = request.FILES['decFile']
            BASE_DIR = Path(__file__).resolve().parent.parent
```

```
            fs = FileSystemStorage()
            filename = fs.save(decFile.name, decFile)
            url = fs.url(filename)
            url1 = str(BASE_DIR)+url
            img = Image.open(url1, 'r')
            decoded_image = decode(img)
            decoded_image.save(url1)
        except:
            request.session['decodeError'] = "Image can't be decoded!"
        return render(request, "decImage.html", {'url': url1})
```

Chapter 6

TESTING

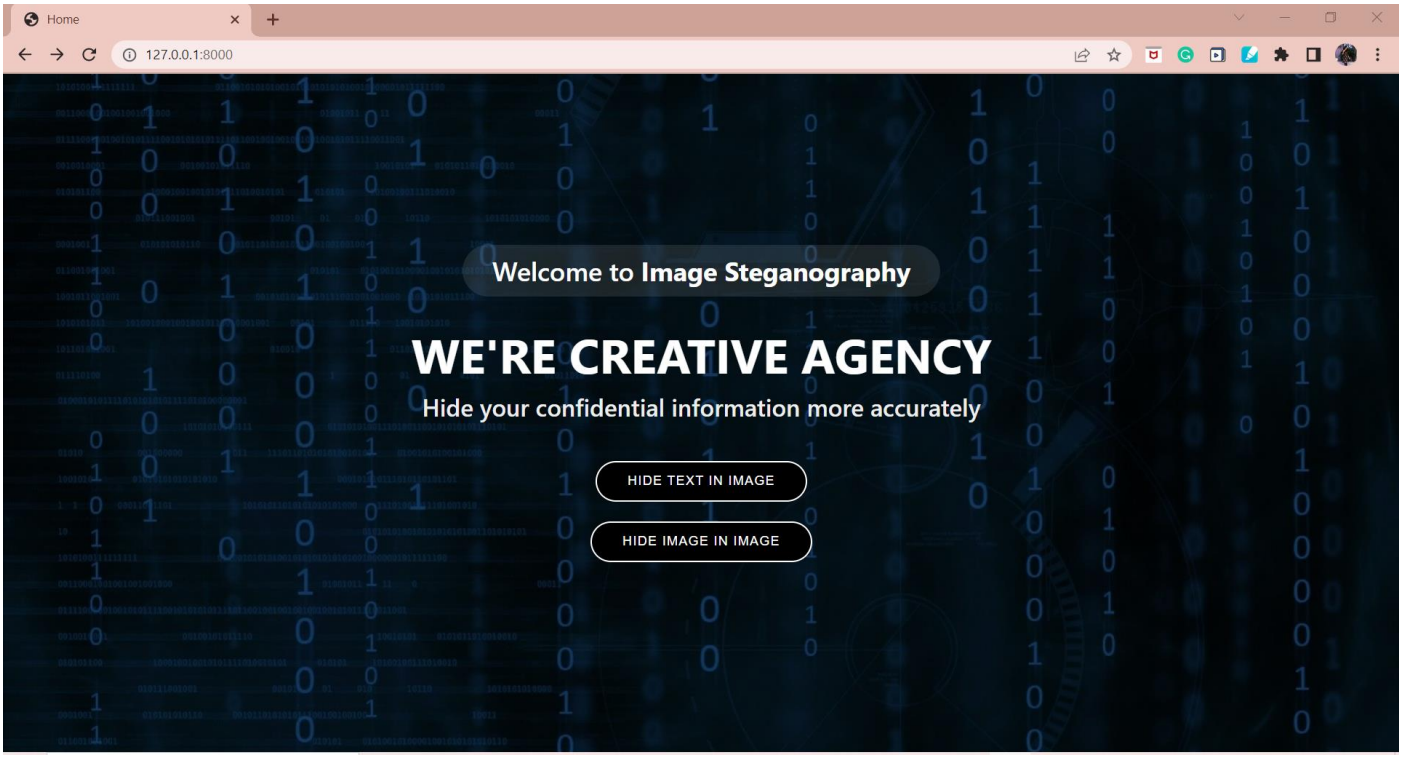
Testing method: Manual testing was performed in order to find and fix the bugs in development process.

Sr no.	Test Scenario	Expected result	Actual result	Status
1	Enter to the website	Home page should be displayed with two buttons for different buttons	Home page is displayed	Success
2	Click on 'Hide text in image' button	Home page of 'hide text in image' module should be displayed	Home page is displayed	Success
3	Length of message is too long to fit in image	'Need larger image size' error is displayed	Error is displayed	Success
4	Encode with appropriate input	Successful encoding message is displayed with various functionalities	Message is displayed	Success
5	Click on 'view image'	Image is displayed in full screen	Image is displayed	Success
6	Click on 'PSNR value'	PSNR value of stego image is displayed	PSNR value is displayed	Success
7	Click on 'download image'	Stego image should be downloaded on users machine	Image is downloaded	Success
8	Decode with correct stego image	Hidden message is displayed on the screen with a back button	Hidden message is displayed	Success
9	Decode with incorrect stego image	'no hidden message' error is displayed	Error is displayed	Success
10	Click on 'Hide image in image' button	Home page of 'hide image in image' module should be displayed	Home page is displayed	Success
11	Encode with inappropriate images	'Need larger image size' error is displayed	Error is displayed	Success
12	Encode with appropriate images	Successful encoding message is displayed with various functionalities (view, psnr, download)	Message is displayed	Success

13	Invalid stego image is provided for decoding	‘Image can’t be decoded’ error is displayed	Error is displayed	Success
14	Valid stego image is provided for decoding	Hidden image is displayed with view and download option.	Hidden image is displayed	Success

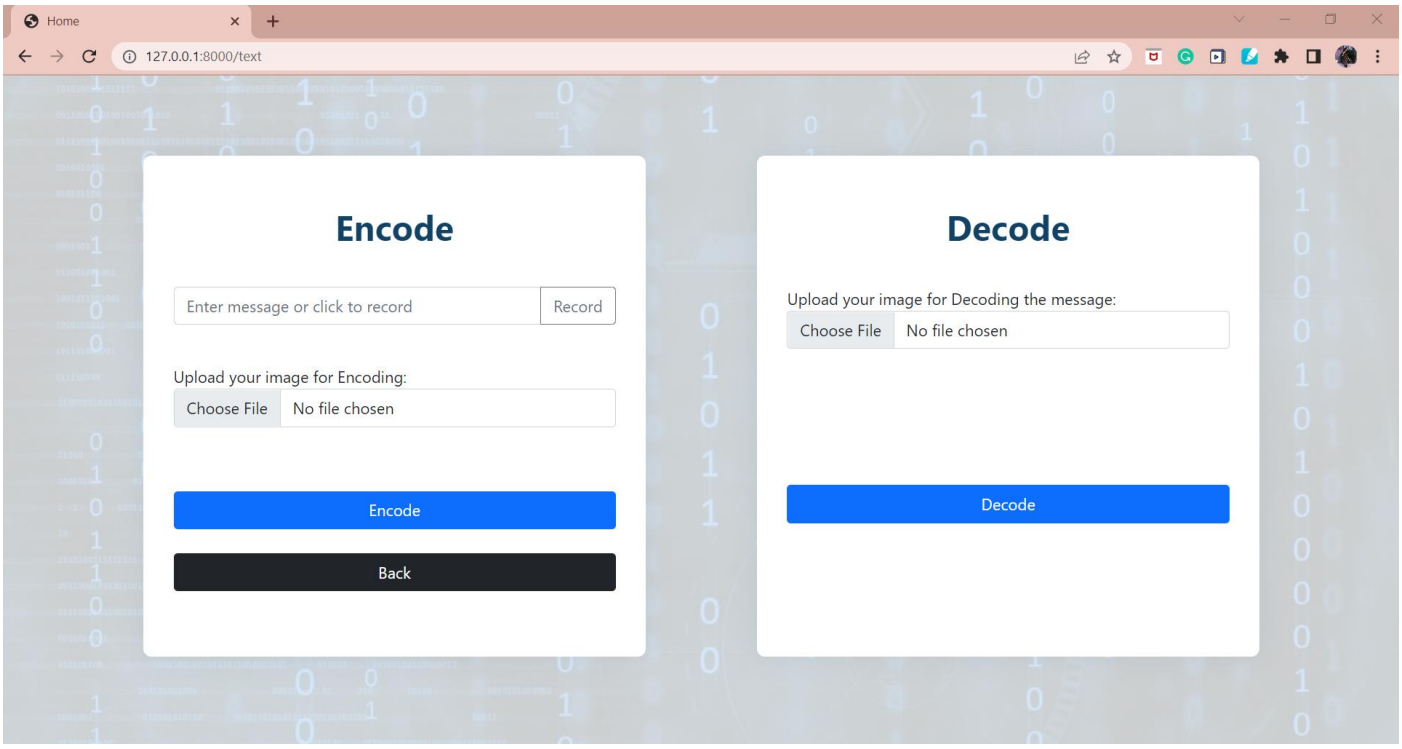
Chapter 7

SCREENSHOTS



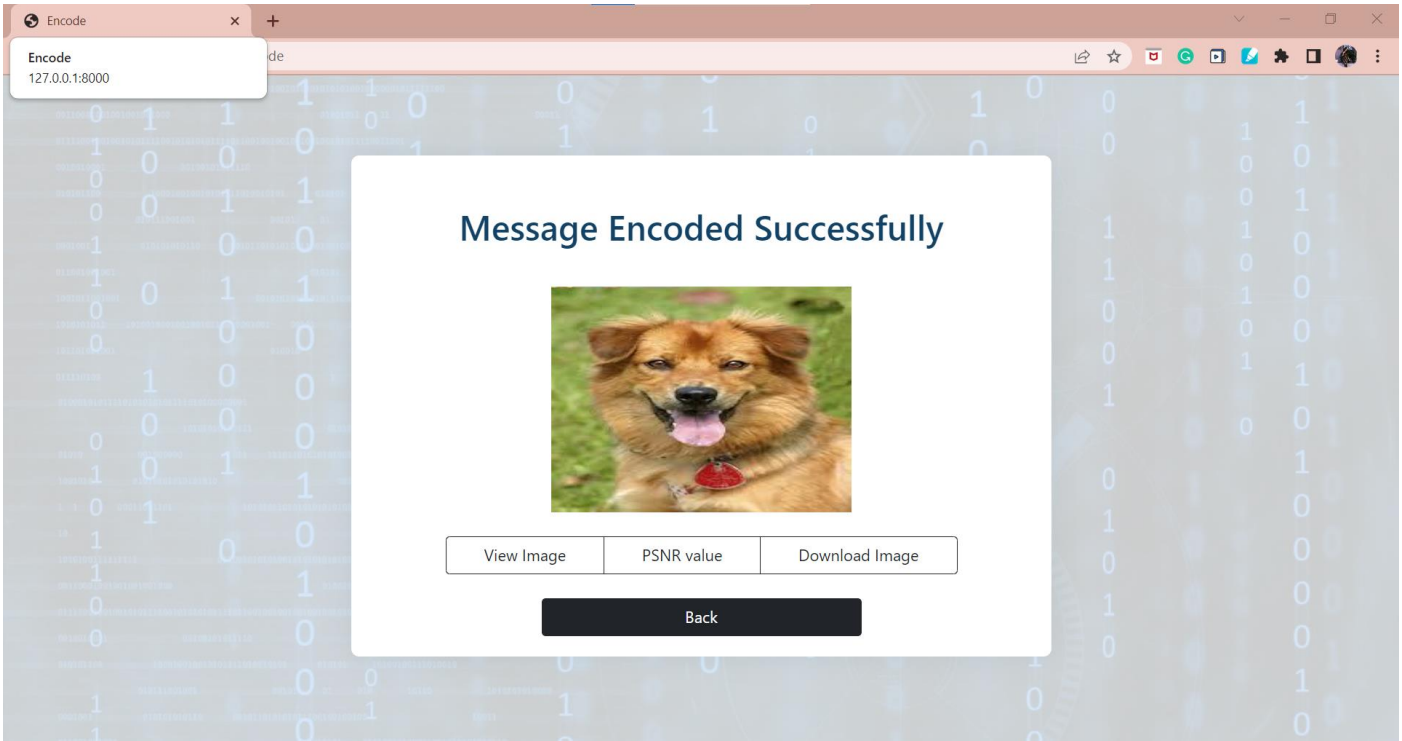
(Figure 7.1) Home page

Figure 7.1 shows the home page of the application.



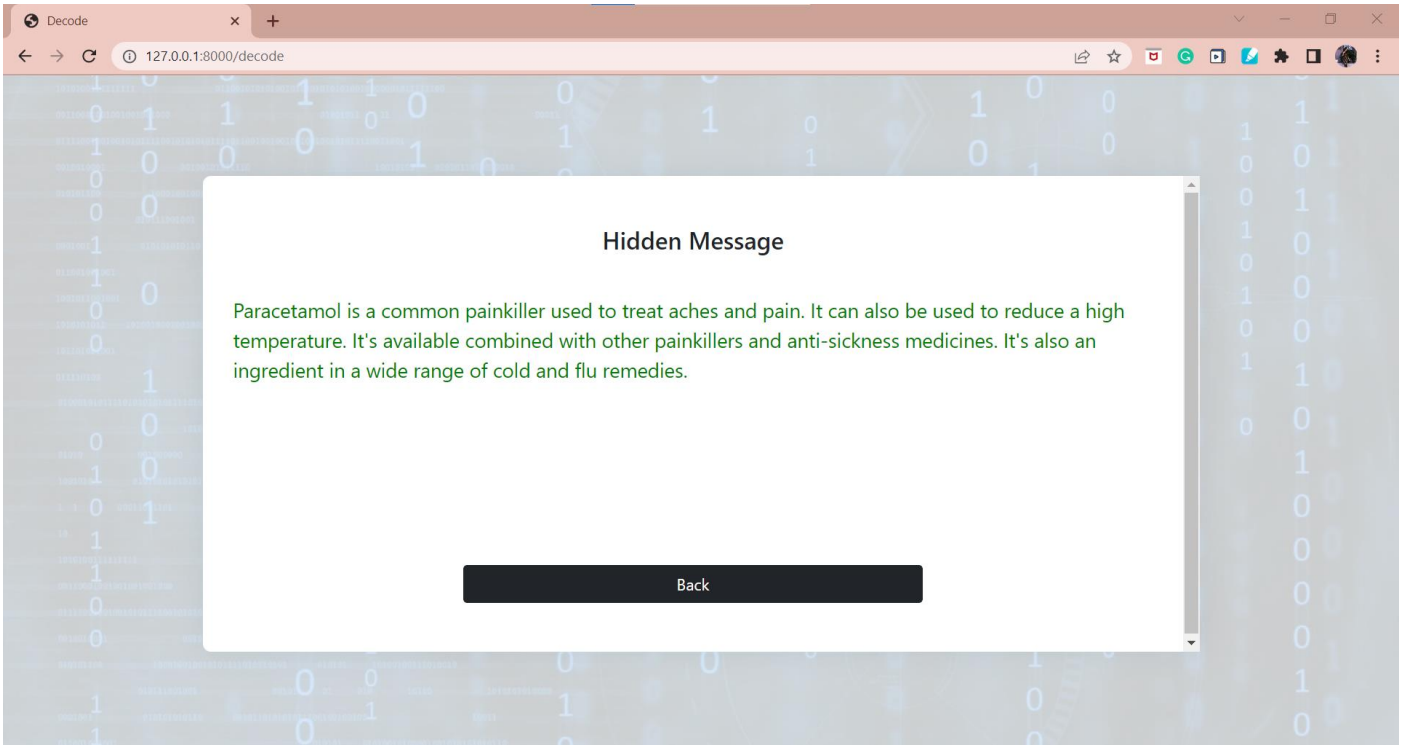
(Figure 7.2) Home page of 'Hide text in image' module

Figure 7.2 shows the encode and decode section of 'hide text in image' module.



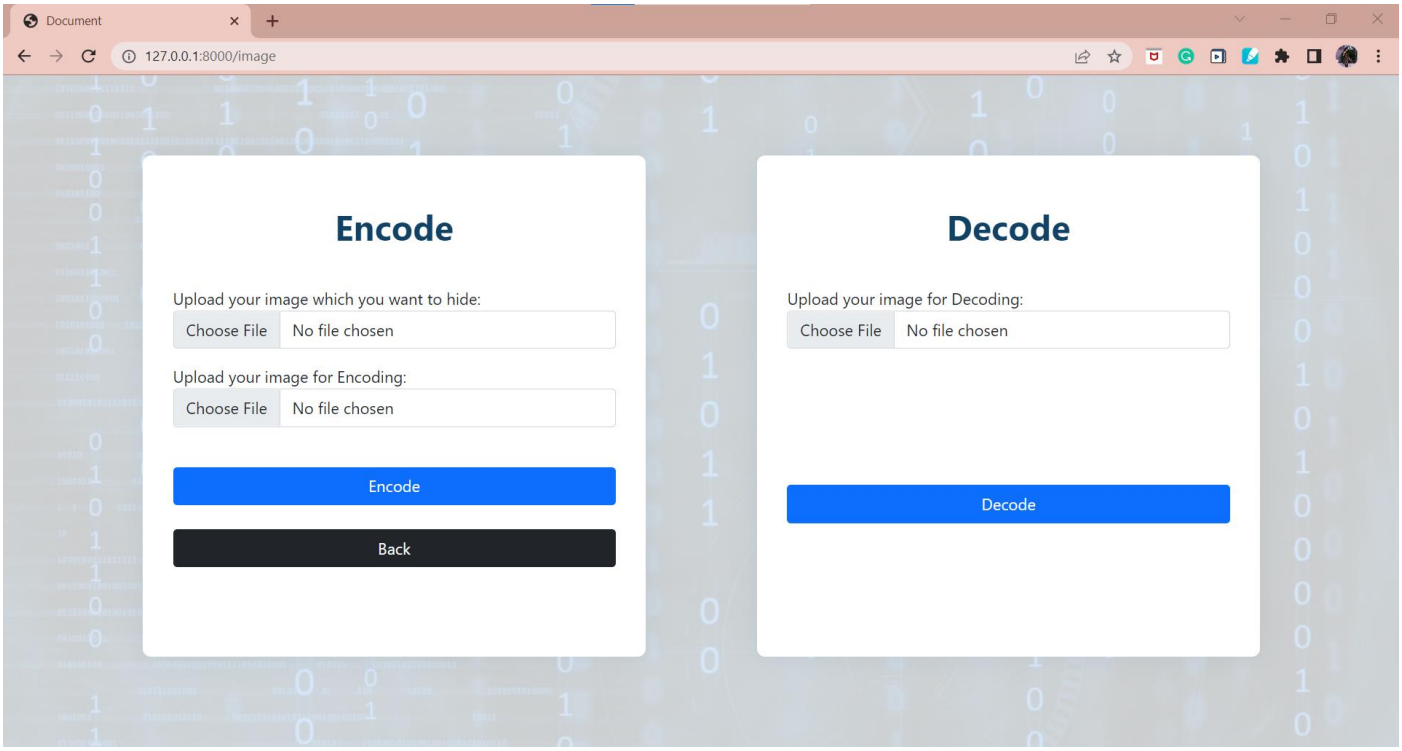
(Figure 7.3) Encode page of 'Hide text in image' module

Figure 7.3 shows if the secret text is hidden in the source image successfully or not. There are several options to view or download image and check its PSNR value.



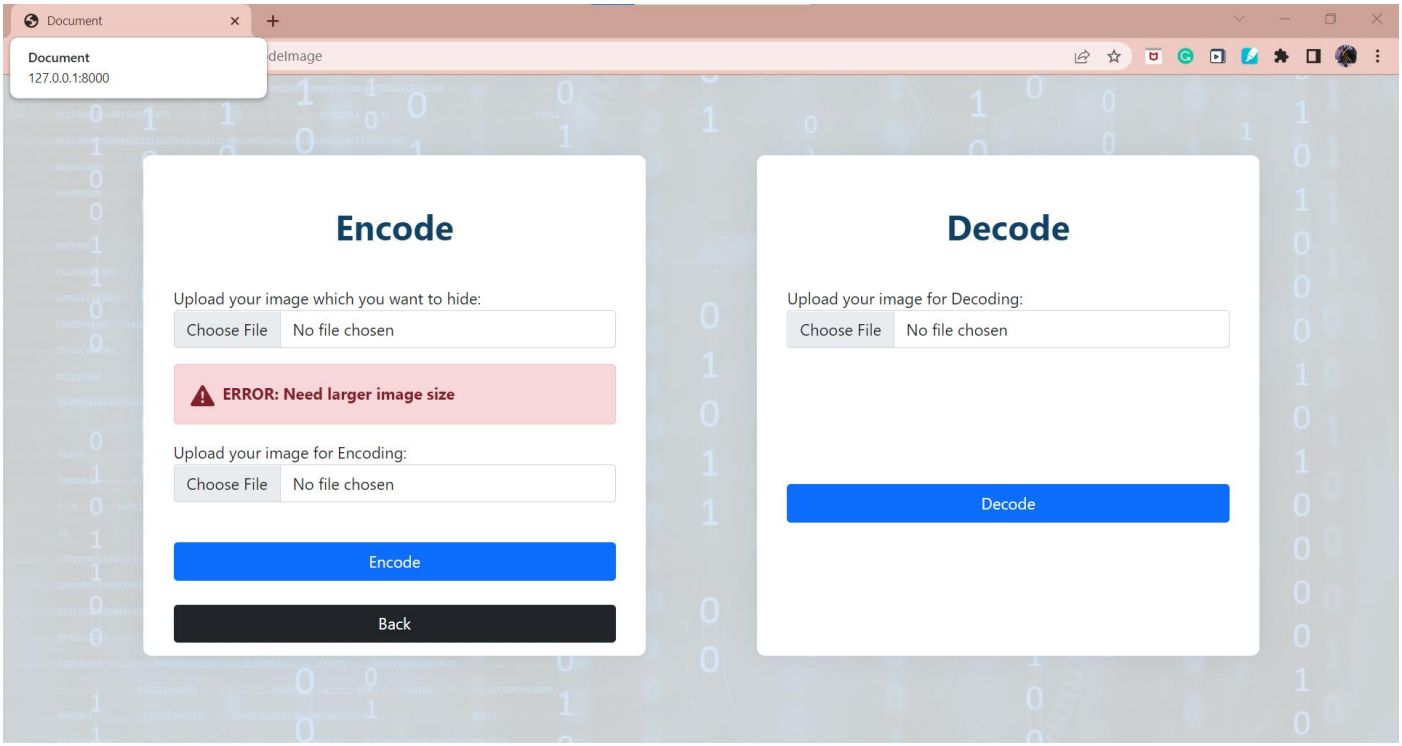
(Figure 7.4) Decode page of 'Hide text in image' module

Figure 7.4 decodes the secret text from the stego image and displays it. If not, appropriate error message would be displayed.



(Figure 7.5) Home page of 'Hide image in image' module

Figure 7.5 shows the encode and decode section of 'hide text in image' module.

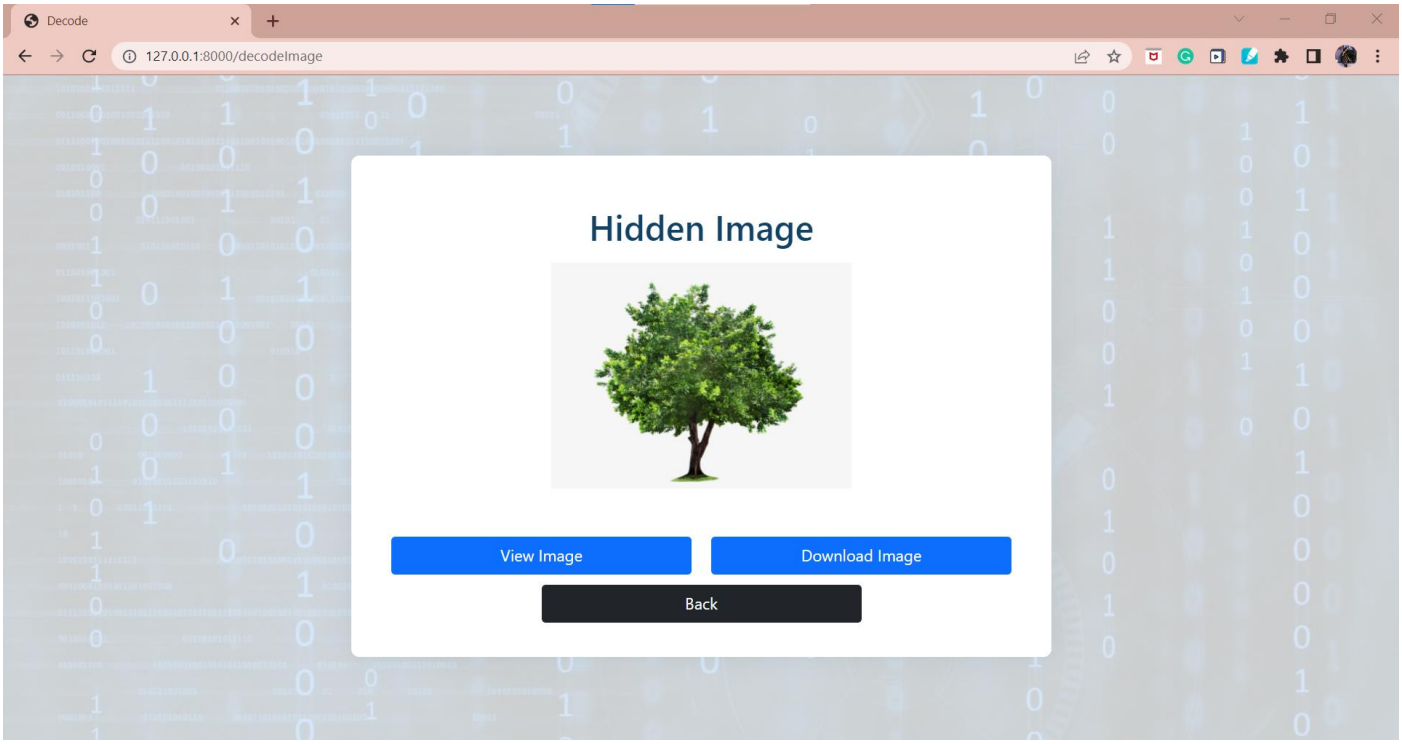


(Figure 7.6) Image error

Figure 7.6 displays error message if the image of appropriate size is not entered.

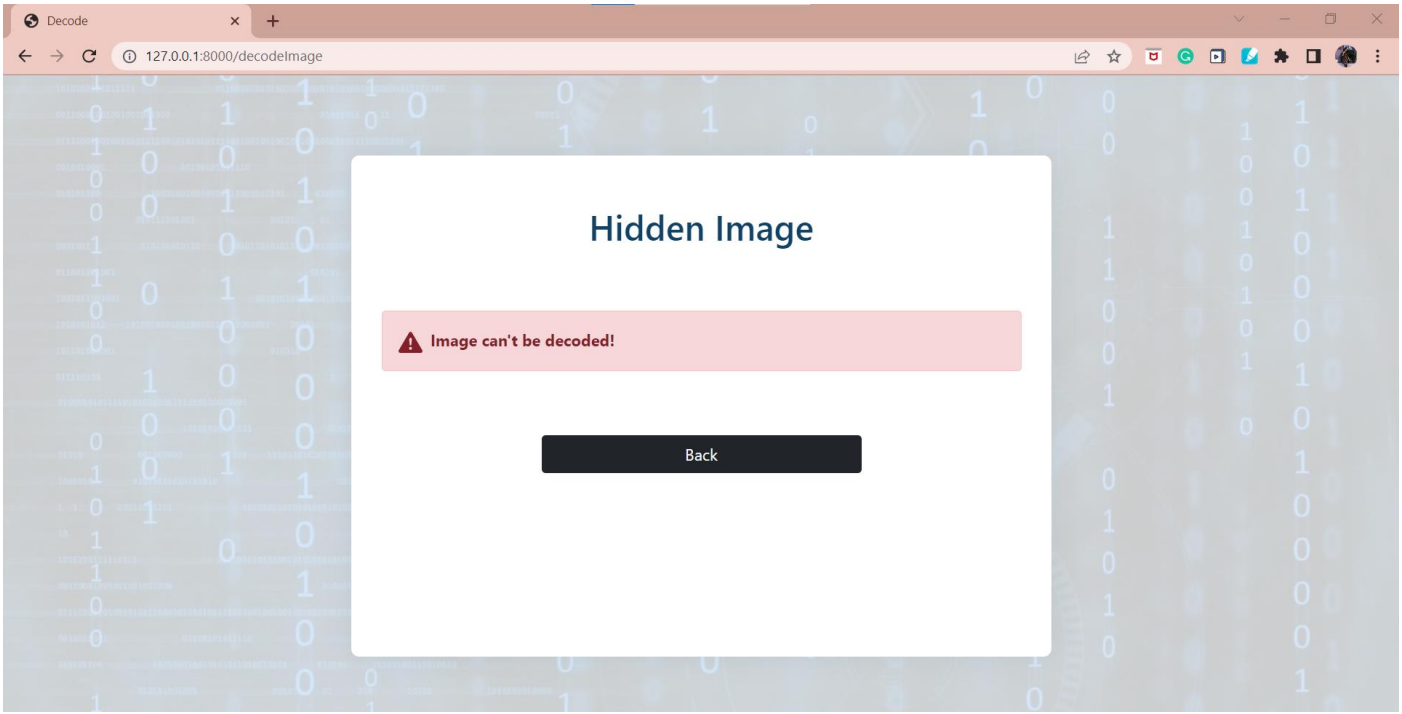
(Figure 7.7) Encode page of 'Hide image in image' module

Figure 7.7 shows the encode page after encoding the secret image into the cover image successfully.



(Figure 7.8) Decode page of 'Hide image in image' module

Figure 7.8 displays the hidden image from the stego image if it is present, if not, error message would be displayed. Options to view image and download image are also available.



(Figure 7.9) Decode error

Figure 7.9 displays error if the stego image cannot be decoded.

Chapter 8

CONCLUSION

The functionalities that are implemented in the system are prepared after understanding all modules according to software requirements specifications (SRS). Functionalities that are successfully implemented in the system are as follows:

- Encoding text in cover image
- Decoding text from stego image
- Encoding image in cover image
- Decoding image from stego image

It is observed that through LSB substitution steganographic method, the results obtained in data hiding are pretty impressive as it utilizes the simple fact that any image could be broken up to individual bit-planes each consisting different levels of information.

Chapter 9

LIMITATIONS AND FUTURE EXTENSIONS

- **Limitations:**

Stego image can have attacks like diluting, noising or contrast changes. If more than two people have same steganography software then the hidden message can be decode successfully. If the PSNR value of stego image is very low then the hidden message can be suspected by any intruder.

- **Future extensions:**

Image steganography is fairly a new idea. There are certain extensions that can be implemented in terms of advancement of application. Cryptography and steganography can be used simultaneously to reduce the exposure of the hidden image. The user can be allowed to choose the bits in which the data is supposed to be hidden. This way attacker would not be able to extract data bits from the stego image. The WWW makes an extensive use of inline images. It may be possible to develop an application to serve as a web browser to retrieve data embedded in the web page images.

Chapter 10

BIBLIOGRAPHY

Following links and websites were referred during the development of this project:

"Image Steganography Explained | What is Image Steganography?"

GreatLearning Blog: Free Resources what Matters to shape your Career!,17 Sept. 2020, https://www.mygreatlearning.com/blog/image-steganography-explained/.

"Python | Peak Signal-to-Noise Ratio (PSNR) – GeeksforGeeks."

GeeksforGeeks, 31 Jan. 2020, https://www.geeksforgeeks.org/python-peak-signal-to-noise-ratio-psnr/.

"Steganography Tutorial | A Complete Guide For Beginners | Edureka."

Edureka, 31 Jan. 2019, https://www.edureka.co/blog/steganography-tutorial.