SWE 599 Final Project Report

# An Image Steganography Tool

BounSteg - Reha YILMAZLAR

# Contents

Introduction ..... 3
Image Steganography ..... 3
Least Significant Bit (LSB) ..... 3
Description ..... 7
Requirements ..... 7
Functional Requirement ..... 7
Non-Functional Requirements ..... 7
Design ..... 8
Use Case Diagram ..... 8
Sequence Diagram ..... 9
Class Diagram ..... 10
Interface ..... 11
Implementation ..... 12
ImageProcess ..... 12
is_corrupt(filename): ..... 12
get_image_attributes(filename): ..... 12
convert_to_kilobyte(size): ..... 13
has_magic(): ..... 13
hide_file(filename): ..... 13
hide_text(message): ..... 14
add_secret_message(): ..... 14
check_space(pixels, secret_binary): ..... 14
save_image(): ..... 14
show_image(): ..... 14
EmbedDialog ..... 15
RetrieveDialog ..... 15
MessageBox ..... 15
Demonstration ..... 16
Text Embedding ..... 16
File Embedding ..... 21
Conclusions ..... 25

Cited References ..... 26
Some Additional Information ..... 26
Bibliography ..... 26

## Introduction

The word "steganography" comes from the Greek language, and it means secret/covered writing. It is used to hide messages in different mediums such as video, image, and audio in a digital world.

The increase in usage of messaging applications and how they store and transfer data between nodes are getting more questionable every day. Recent data leaks, hackings in big tech companies show us that cybersecurity should be and will be a more important aspect of our lives.

Steganography is a part of cybersecurity. Back then terrorists, spies, hackers used steganography to transfer secret messages in plain sight. They were and still are probably hiding messages in places that we visit every day but do not pay attention enough. Nowadays, you could even upload a picture to one of your favorite social media platforms with a secret message embedded, and probably no one would even think that it might include a secret message. It could be anywhere, in a video, music file, image file, and so on.

## Image Steganography

With the increase of cybersecurity attacks that came with the pandemic, people are being trained against those attacks a lot more. So that, steganography methods are used more than ever probably. The most used kind of steganography in those kinds of cybersecurity training is image steganography. In this kind of steganography, an image is used as the cover media.

Virtual machines are deployed with a few security flaws to train and educate people against those vulnerabilities. First, you hack the system then you know what the flaws and/or misconfigurations are so you can defend properly in the future. Hacking is gamified in a way because those machines do not actually have a real website that real people use. They are used to show security flaws and what might happen if we do not fix them. Image steganography is widely used on virtual machines to deliver a secret hint, hide a file or a password in an image.

### Least Significant Bit (LSB)

LSB is one of the most common image steganography methods and one of the simplest ones. It is used to embed a secret message by using the least significant bit. In an image file, the last bits of the RGB values of a pixel are simply changed to hide data.

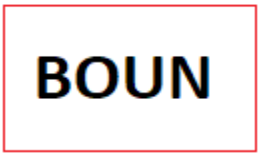



*Fig.1* The word "BOUN" written with Calibri Font

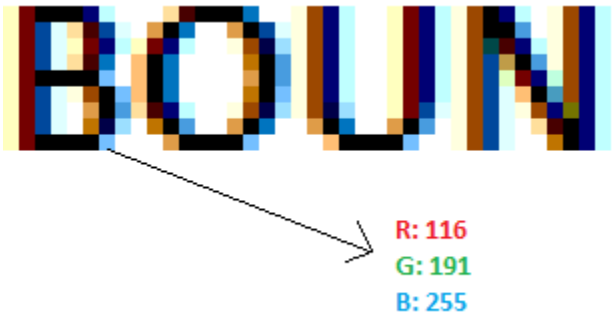



*Fig.2* Figure 1 Zoomed In

As you can see above, all images that are stored digitally, consist of pixels. Those pixels hold the red, green, blue values (RGB). Usually, those color levels are stored from 0 to 255 in 8 bits. (00 and FF in hexadecimal notation.) Thus, the computer knows which color that pixel has. When those pixels get together, the image is created.

Let's break down how our secret data will be stored in the image, without any significant change to the eye.

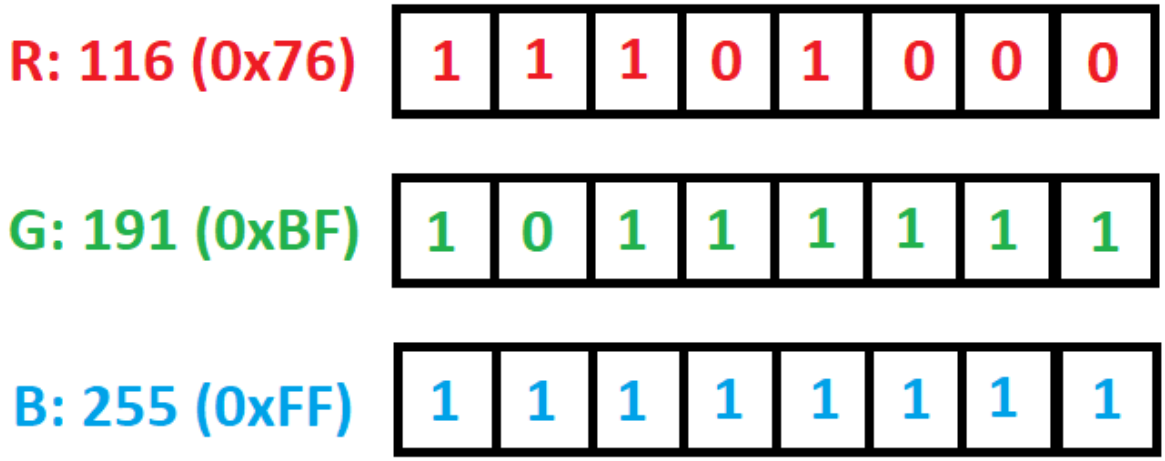



*Fig.3* RGB Values into Bits

The image above shows RGB values converted into binary for only one pixel. Fortunately, images have hundreds or thousands of them.

If we take the 6 following pixels next to our chosen one, we get:

[(116, 191, 255), (255, 255, 255), (255, 255, 255), (255, 255, 255), (223, 156, 72), (0, 0, 0) (0, 0, 0)]

If we write them in binary:

116 = $1110100_2$

196 = $1110100_2$

255 = $1111111_2$

255 = $1111111_2$

255 = $1111111_2$

255 = $1111111_2$

-

-

-

0 = $00000000_2$

Now it's time to hide our secret message. "Hi" will be the message we will hide in those pixels.

If we convert it into binary (UTF-8 encoded):

H = $01001000_2$

i = $01101001_2$

After that, we will change the least significant bits in pixels.

Take 116 = 1110100, the last bit here is the least significant bit. If we take one bit from the letter H's binary value and swap it with the pixel's least significant bit, we will have:

H = $01001000_2$

First pixel,

116 = $1110100_2$ (remains unchanged)

197 = $1110101_2$ (changed)

254 = $1111110_2$ (changed)

Second pixel,

254 = $1111110_2$ (changed)

255 = $1111111_2$ (remains unchanged)

254 = $1111110_2$ (changed)

Third pixel,

254 = 1111110$_2$ (changed)

255 = 1111111$_2$ (remains unchanged)

254 = 1111110$_2$ (the letter "i" will start from this pixel's blue value)(changed) i = 01101001$_2$

Fourth pixel,

255 = 1111111$_2$ (remains unchanged)

255 = 1111111$_2$ (remains unchanged)

254 = 1111110$_2$ (changed)

Fifth pixel,

223 = 11011111$_2$ (remains unchanged)

156 = 10011100$_2$ (remains unchanged)

72 = 1001000$_2$ (remains unchanged)

Sixth pixel,

255 = 1111111$_2$ (remains unchanged)

255 = 1111111$_2$ (we do not need those values, we have "i" letter stored already)

255 = 1111111$_2$ (we do not need those values, we have "i" letter stored already)

As a result, with only 6 pixel's RGB values, we could store the "Hi" word and we just changed 6 pixels' 9 color values by 1 and left the remaining 9, as it was. 50% of the values are unchanged.

Let us see the new colors.

*Fig.4* Pixels that will be changed.

*Fig.5* Pixels that are changed

As seen, there is no visible change to the human eye and it usually depends on the image and how many bits are changed. With the help of this method, lots of data can be stored with low computational time complexity. The downside of this method is that it is fairly easy to detect with existing steganography analyzing algorithms.

## Description

Python and image module pillow (PIL) are used. It is a Windows application with a graphical interface that is created with pyqt5. PyInstaller is used which helps to convert any python script file (.py file) into windows executable file by packing necessary python and modules into one file.

It is deployed as windows executable file (with .exe extension). Python script is kept for other use cases. (Python script for MacOS, Linux).

BMP and PNG extensions are supported at the moment.

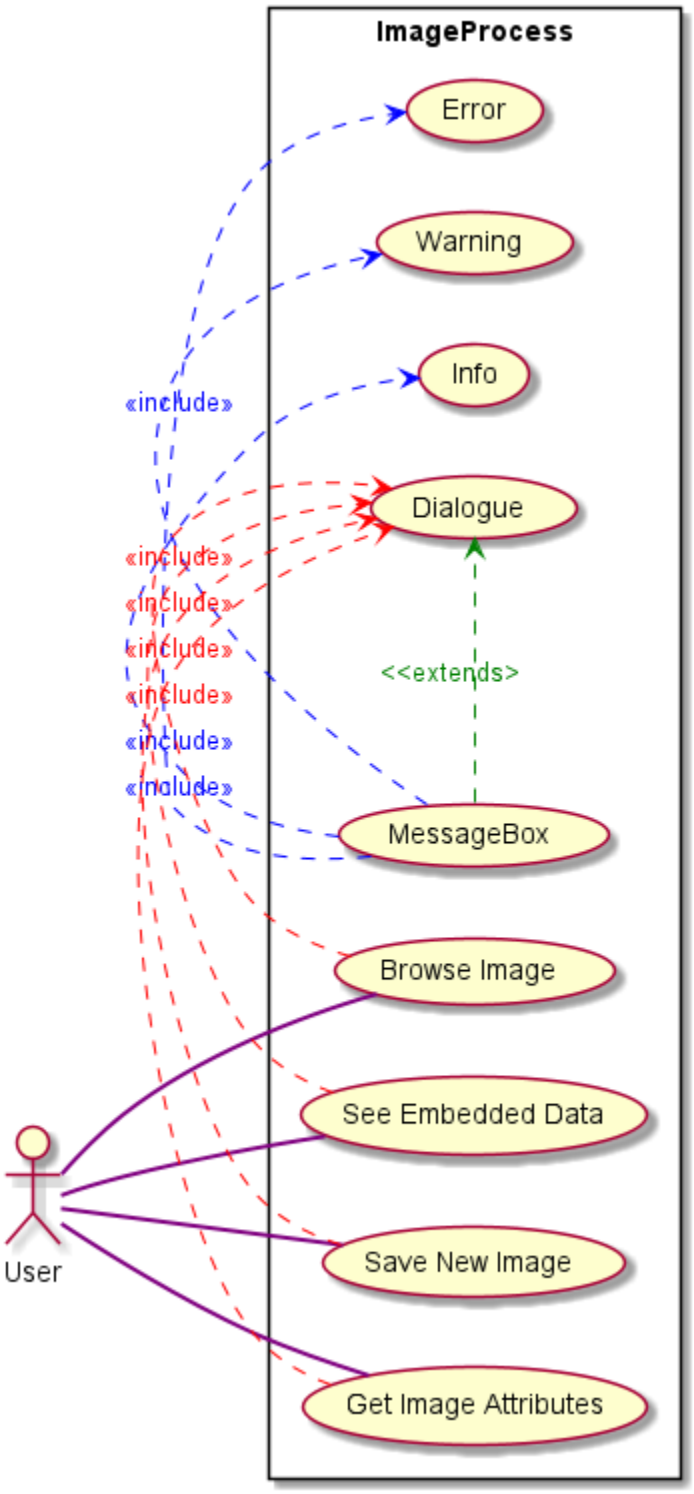
## Requirements

| No. | Functional Requirement |
|---|---|
| 1 | User will choose an image file to work on. |
| 2 | User will choose a text or a file to embed into the desired image file. |
| 3 | Application will create the new image with the hidden text, or the file and user will save it wherever they want. |
| 4 | Application will tell the user if the file or text input size is small enough to be stored in the chosen image. |
| 5 | After embedding the data into a newly created image, user can extract the image with the same program. |

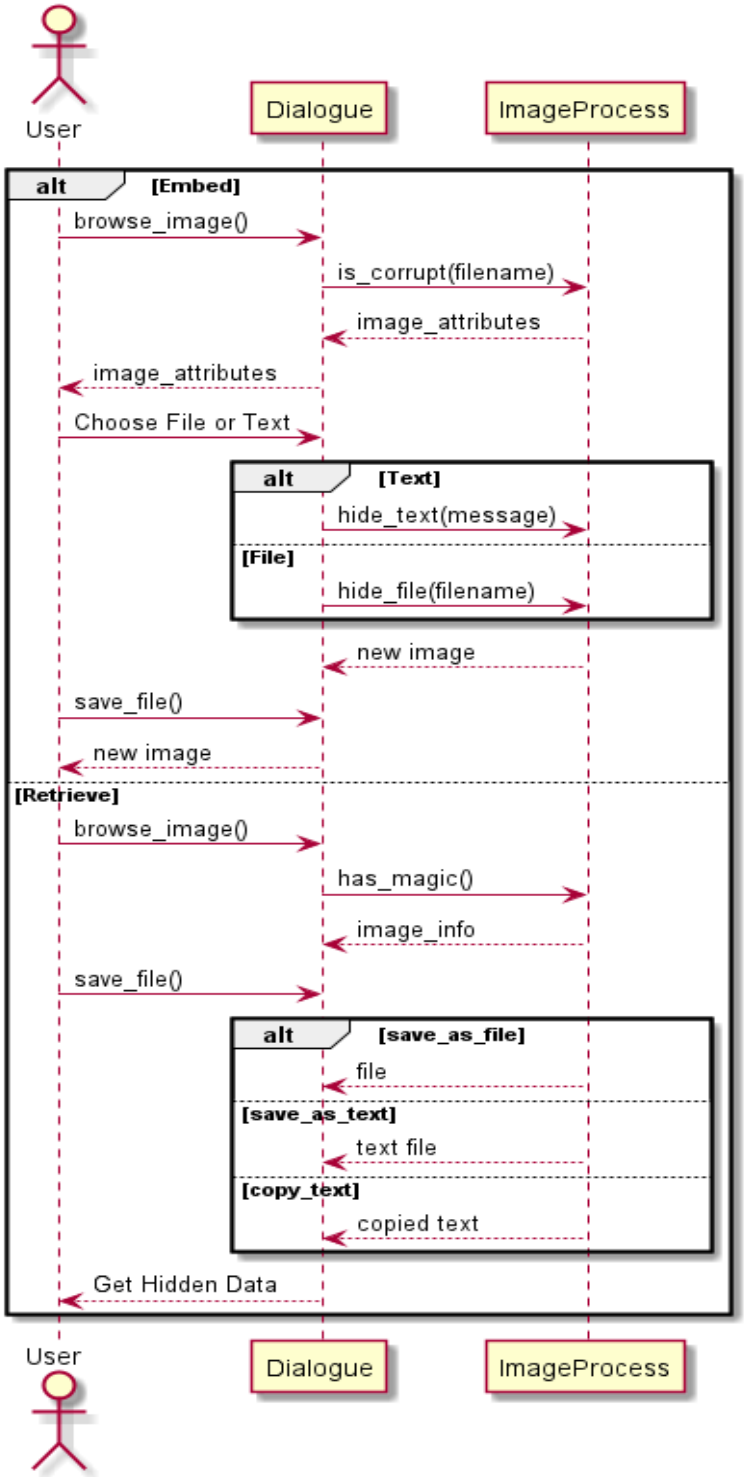
| No. | Non-Functional Requirements |
|---|---|
| 1 | Application shall be a Windows executable. |
| 2 | It shall have a simple graphical interface. |
| 3 | User will choose images with BMP or PNG extensions to work on. |
| 4 | As the programming language Python will be used. |
| 5 | Application shall be scalable and maintainable. |
| 6 | It shall run in any Windows 10 device after necessary modules are installed. |
| 7 | Application will show some simple information about chosen image. |
| 8 | It will be a desktop application. (With a graphical interface.) |

# Design

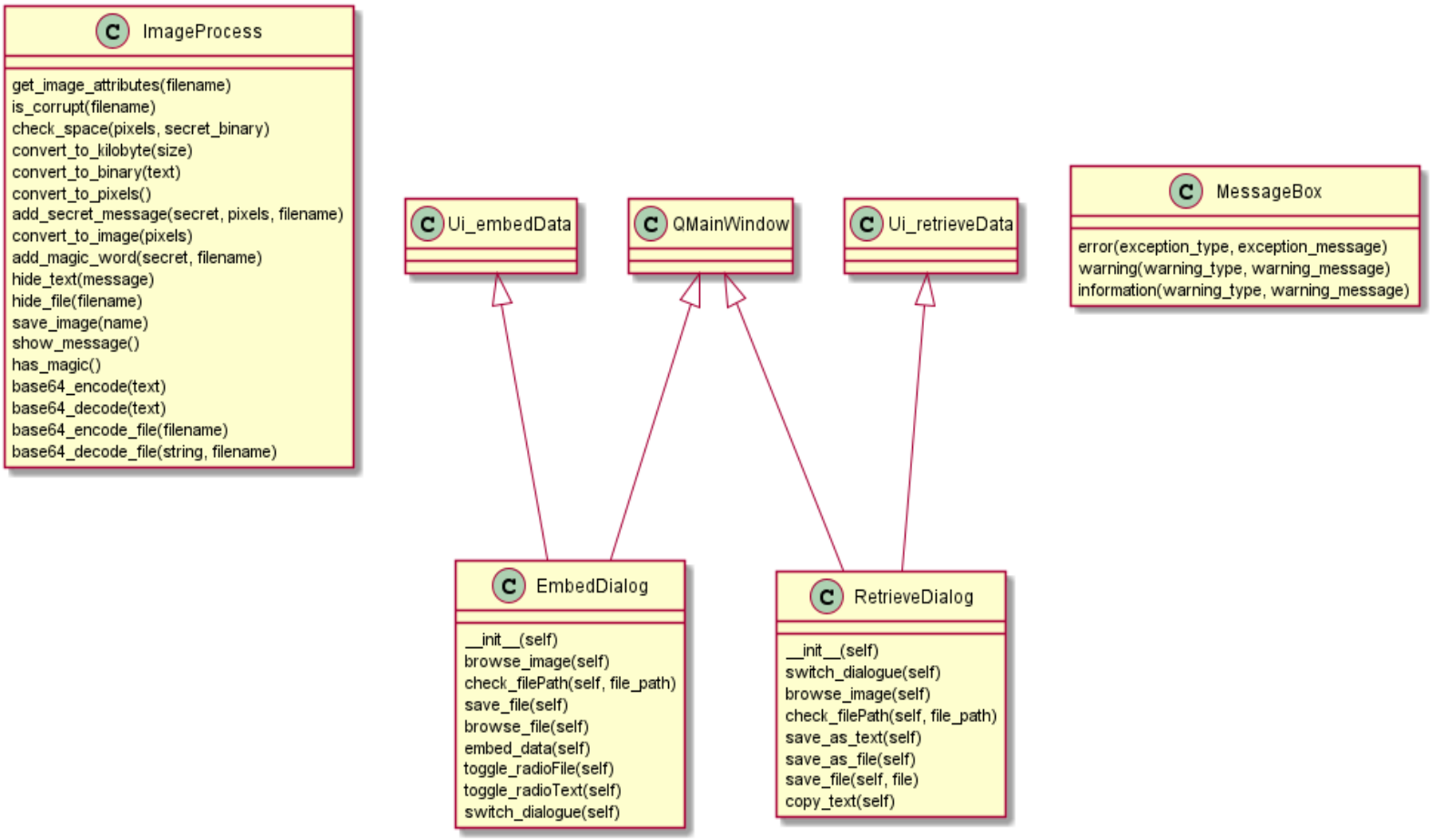
## Use Case Diagram



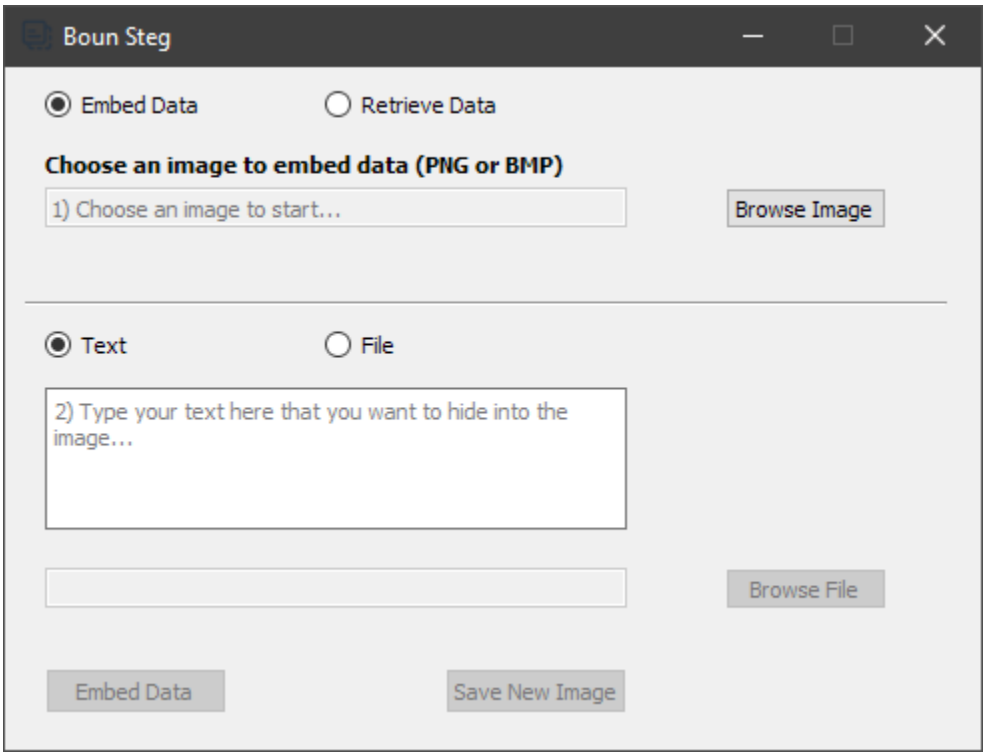
## Sequence Diagram

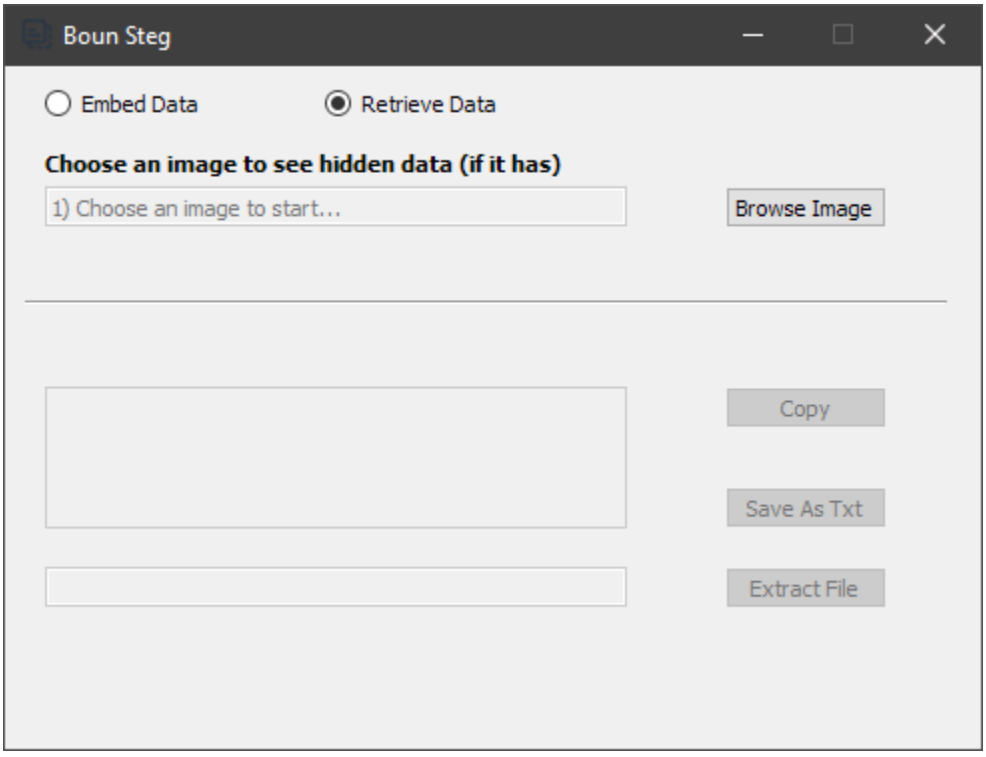


## Class Diagram



## Interface


Boun Steg
Embed Data
Retrieve Data
Choose an image to embed data (PNG or BMP)
1) Choose an image to start...
Browse Image
Text
File
2) Type your text here that you want to hide into the image...
Browse File
Embed Data
Save New Image


Boun Steg
Embed Data
Retrieve Data
Choose an image to see hidden data (if it has)
1) Choose an image to start...
Browse Image
Copy
Save As Txt
Extract File

## Implementation

There are three main classes in the application:

- ImageProcess
- EmbedDialog
- RetrieveDialog

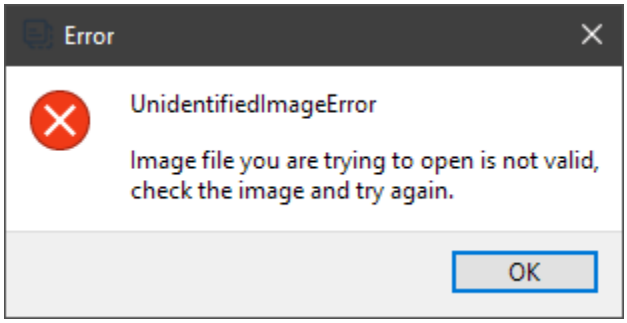
### ImageProcess

This is the backend part of the application that covers the image to the pixel conversion process, turning data/text into binary, appending it to image pixel matrix, and then converting it back to image. It also checks for corrupt image files, and it has a save function that is used to save the new image file. In a nutshell, this class covers all low-level processes and does nothing else.

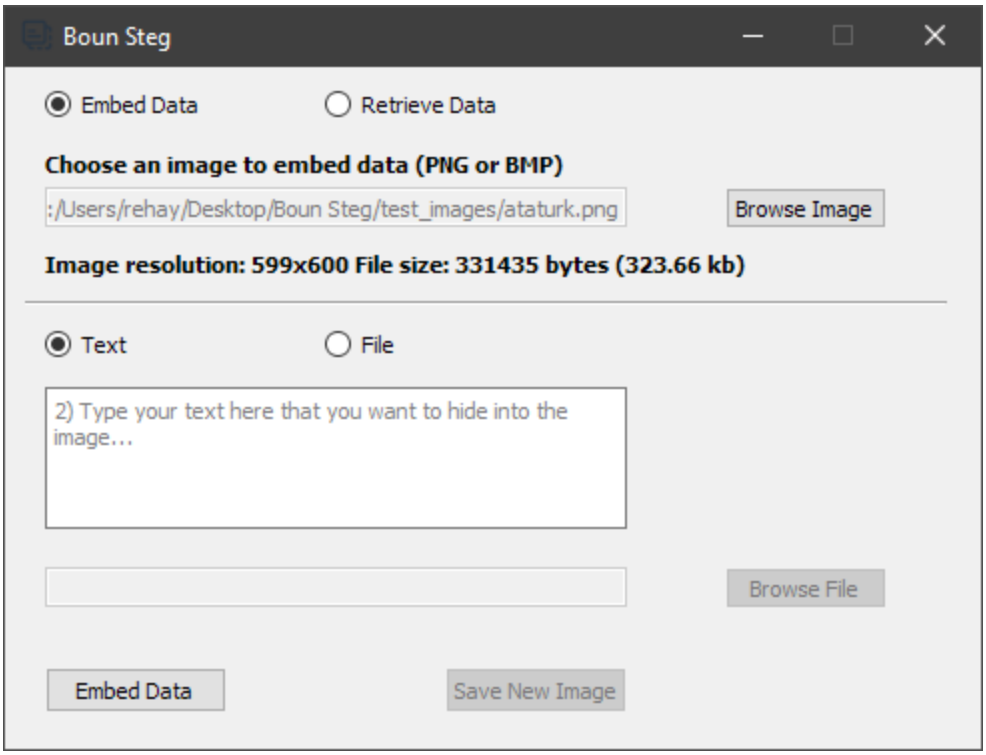
ImageProcess class works together with Embed and Retrieve dialogue classes which form the graphical user interface (GUI) of the application.

After choosing an image to embed text or a file inside, the flow of the ImageProcess functions are as follows:

**is_corrupt(filename):** This method takes the file checks if it is a proper image file then converts it to RGB or RGBA values depending on the format of the file. If it is a PNG image it converts it to RGBA (RGB with alpha channel), otherwise if it is a BMP image it converts it to an RGB array. If file is corrupt, it raises an exception and shows a message box saying so.




**get_image_attributes(filename):** This method simply returns the file attributes as a string in the format of "Image resolution: {width}x{height} File size: {size} bytes ({size_kilobytes} kb)". It converts size from bytes to kilobytes with the help of convert_to_kilobyte(size) function. After, EmbedDialog class takes this value and shows it in the interface.




**convert_to_kilobyte(size):** Simply converts bytes to kilobytes with the precision of 2 decimal points.

**has_magic():** Takes global image_rgb matrix then calls the convert_to_binary function. After convert_to_binary function converts the file or the text to binary, if the image has alpha channel (RGBA) it checks first 12 RGBA values which is 12*4 = 48 bytes, if it does not have alpha channel (e.g., 24 bit – BMP image) it checks first 16 RGB values which is 16*3 = 48 bytes again. After gathering first 48 bytes, it controls if the byte stream has constant "$ß0UN$" or constant "$Reh@$" words. If it has "$ß0UN$" then it knows the image has a hidden file embedded, otherwise, it has a text. If it does not have either of the words in that way it knows there is no secret embedded in the image file.

According to secret data, whether it is a file or text following methods are called:

**hide_file(filename):** Firstly, this method stores the file name that will be embedded in a variable. Secondly, converts the image to pixels with the help of convert_to pixels() method. Thirdly, it encodes the file with Base64 algorithm with UTF-8 encoding so that it does not lose any data because of encoding problems of languages between conversions. Then, it calls add_secret_message method that appends the word "$ß0UN$", filename, "#Reh@#", file data and "#Reh@#" at the end as a stopper word. The stopper word is used twice. First to catch the

filename and second to catch where secret data ends in the image file, so if it ends early the program stops to check the remaining pixels for the data. Finally, everything is converted to binary and is appended from the beginning of the file changing every 8th bit of RGB values.

**hide_text(message):** Basically, it works the same way as the hide_file method, except; this one takes text instead of a file, base64 encodes it and appends the magic_word ("$Reh@$") and stopper_word ("#Reh@#") at the beginning and the end of the constructed string, after converting string to binary, it appends binary string to the beginning every 8th bit of RGB values.

**add_secret_message():** It is called by **hide_text** or **hide_file** functions. It basically appends magic_word, secret data, and stopper word(s) together and returns it as the pixel array. It also calls **check_space(pixels, secret_binary)** method.

**check_space(pixels, secret_binary):** This method is called by add_secret_message method and raises an error if given data does not fit in the image which means there are fewer pixels than the data that is desired to be hidden in the image.

**save_image():** It converts pixels back to an image file.

**show_image():** If has_magic() method returns "File" or "Text" then it means that the browsed image has secret data embedded. Show_message() method looks for the appended magic and stopper words to locate the hidden data. If it finds magic_word ("$Reh@$") it takes every 8th bit until it finds the stopper_word ("#Reh@#"). Then converts it to string. This string then is decoded back to the original text and shown in the interface. However, if the method finds the sequence of "$ß0UN$", then it extracts the file name by looking at the first stopper_word ("#Reh@#"), then extracts the file data by looking at the stopper word a second time. Finally, everything gets converted to a string, then this string base64 decoded back and data is written into a file with the extracted file name.

### EmbedDialog

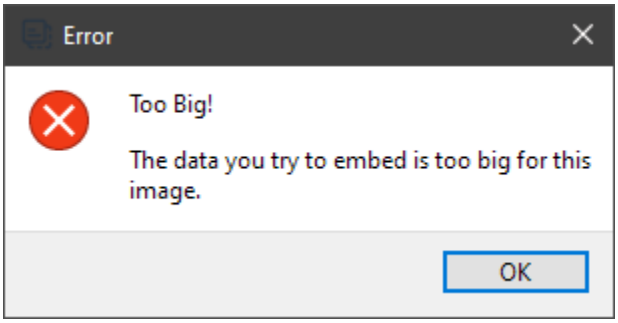
This class is responsible for creating and handling simple graphical interface of the embed part of the application. It simply enables/disables buttons, connects the buttons to ImageProcess class' functions then gets the return value and shows it in the interface.

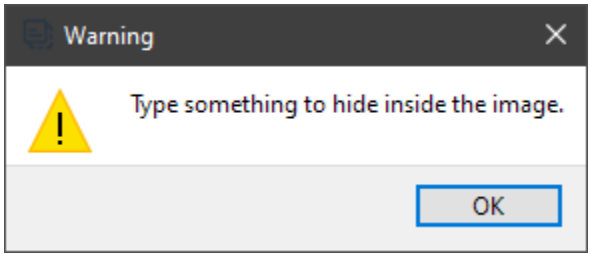
### RetrieveDialog

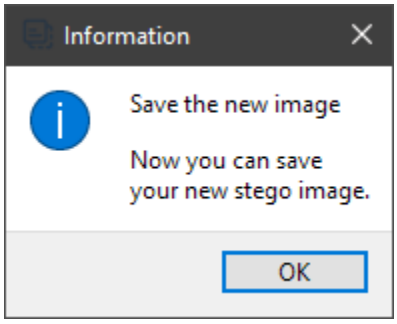
This class handles the retrieval process of hidden data in the graphical interface. Similar to EmbedDialog it handles enabling/disabling of buttons, connecting the buttons to ImageProcess class' functions then showing the return value in the interface.

### MessageBox

This class has three simple methods that construct error, warning or info message boxes as a template. Message and type are passed to methods, then it shows a graphical message box as follows:

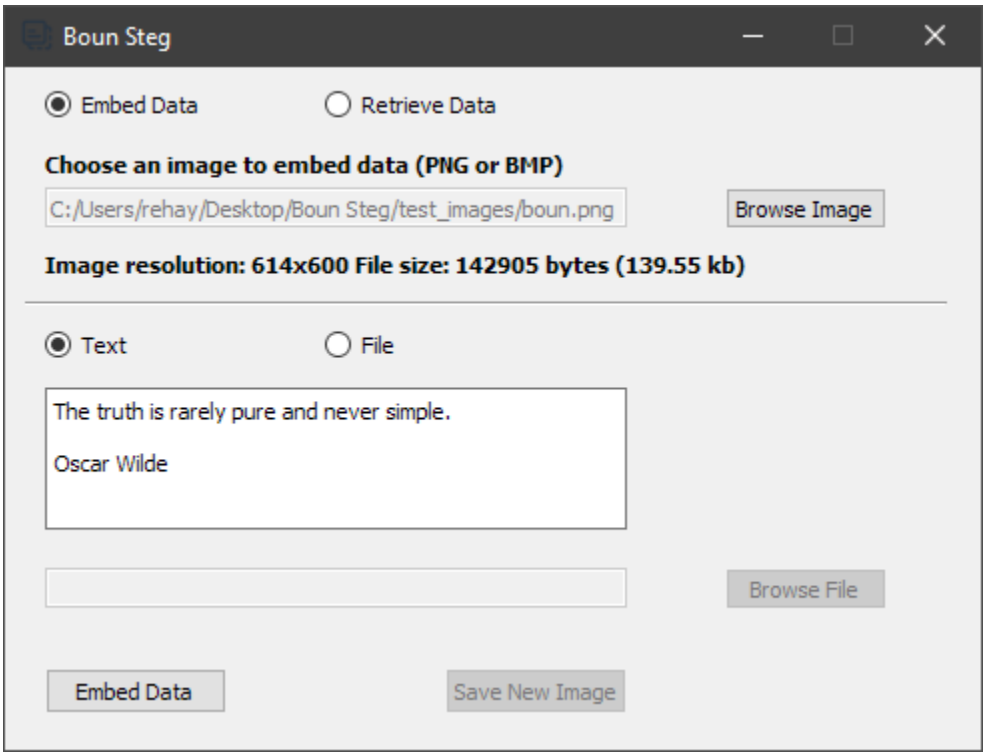


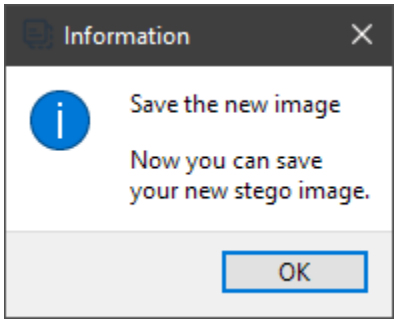



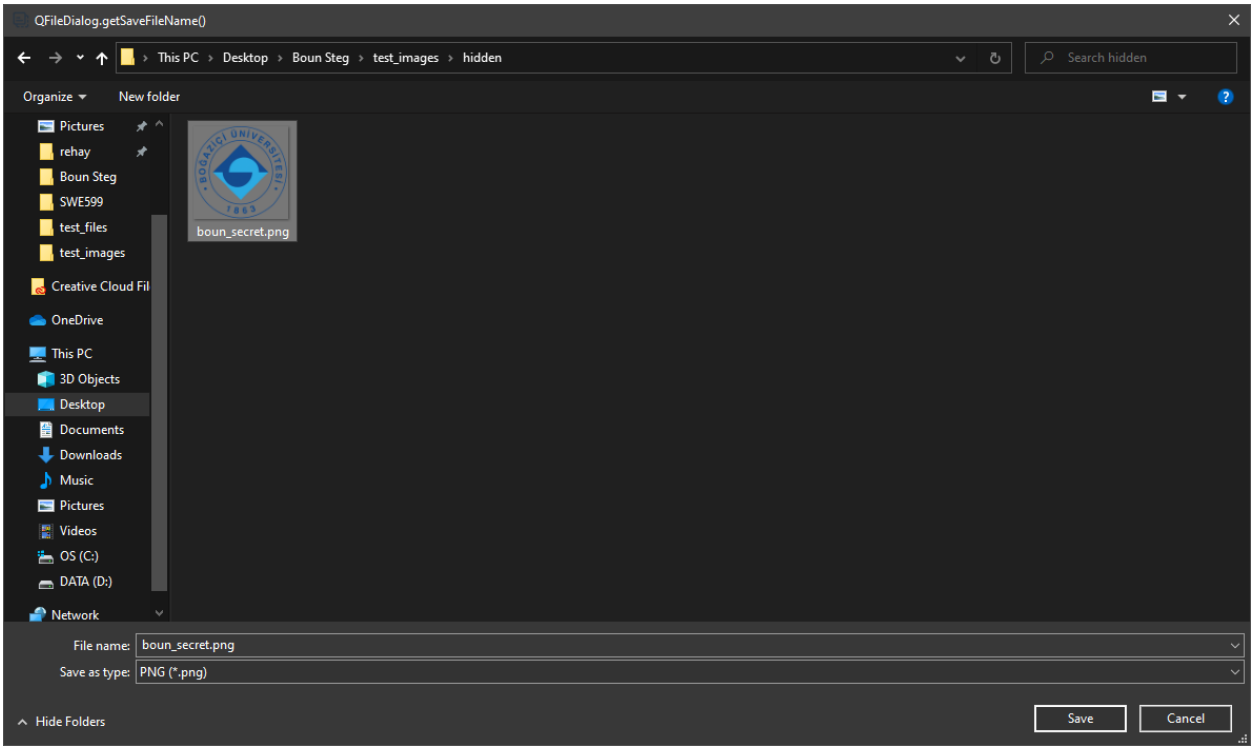



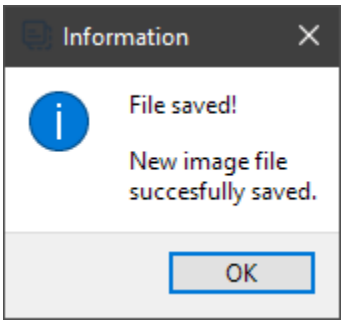
## Demonstration

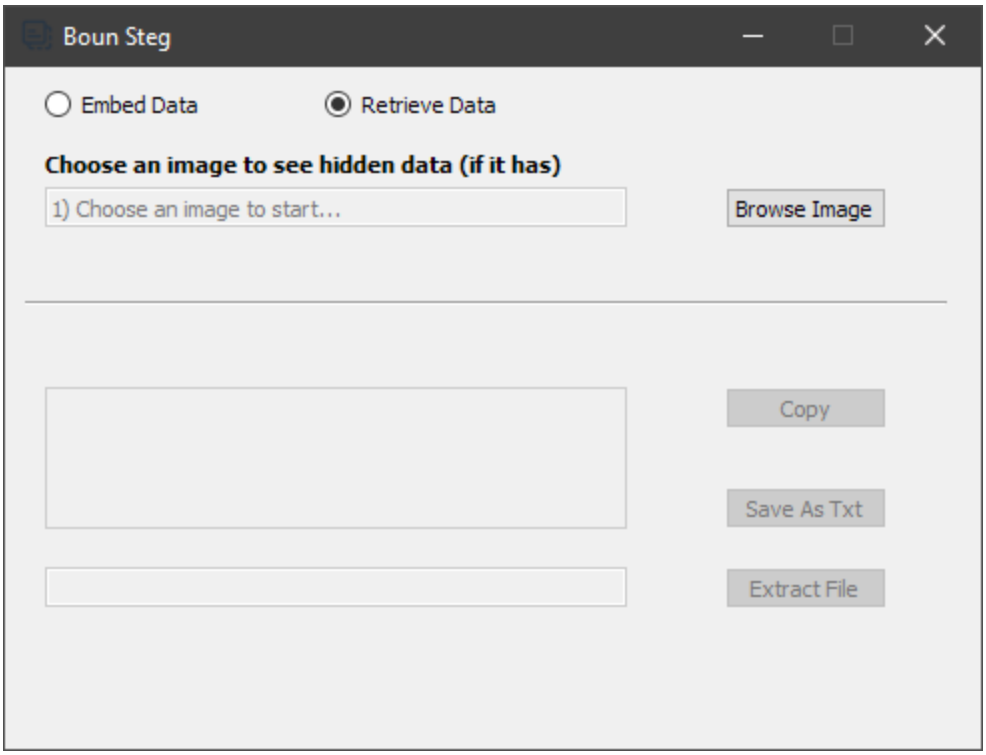
### Text Embedding




QFileDialog.getSaveFileName()
This PC
Desktop
Boun Steg
test_images
hidden
Search hidden
Organize
New folder
Pictures
rehay
Boun Steg
SWE599
test_files
test_images
Creative Cloud Fil
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
OS (C:)
DATA (D:)
Network
boun_secret.png
File name:
boun_secret.png
Save as type:
PNG (*.png)
Hide Folders
Save
Cancel


Information
File saved!
New image file
succesfully saved.
OK


Boun Steg
Embed Data
Retrieve Data
Choose an image to see hidden data (if it has)
1) Choose an image to start...
Browse Image
Copy
Save As Txt
Extract File

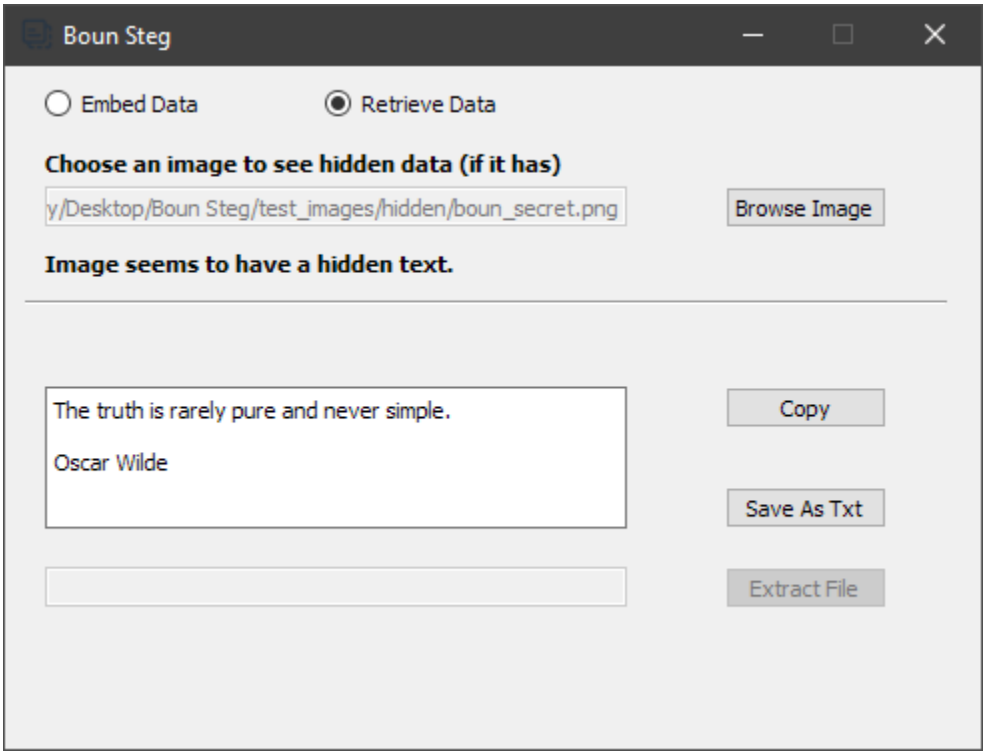
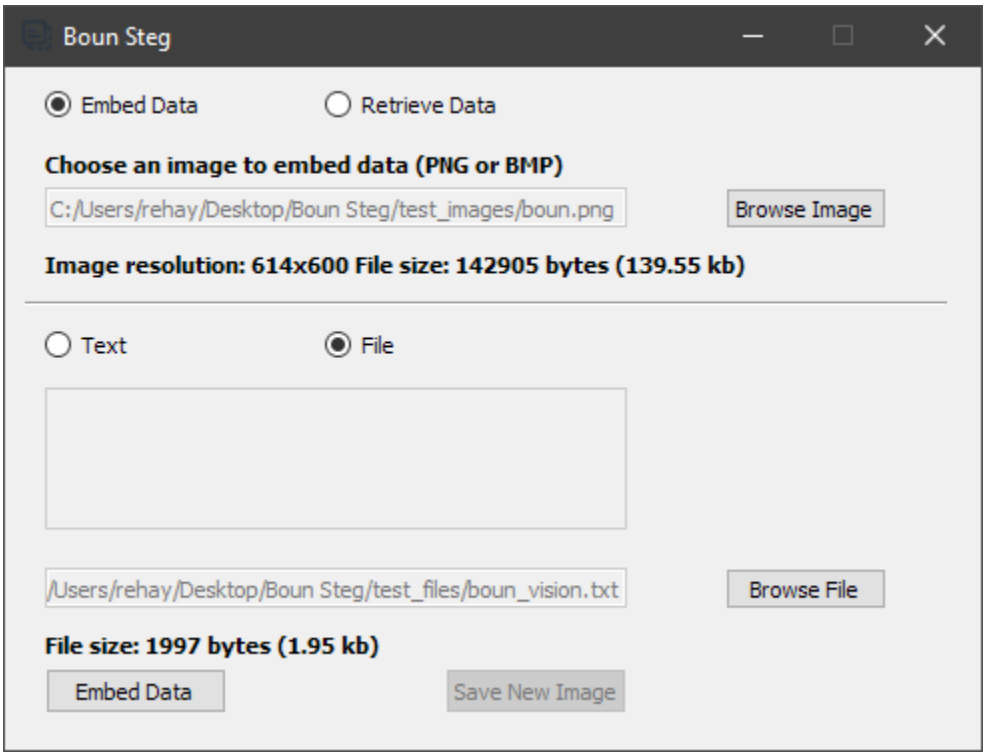
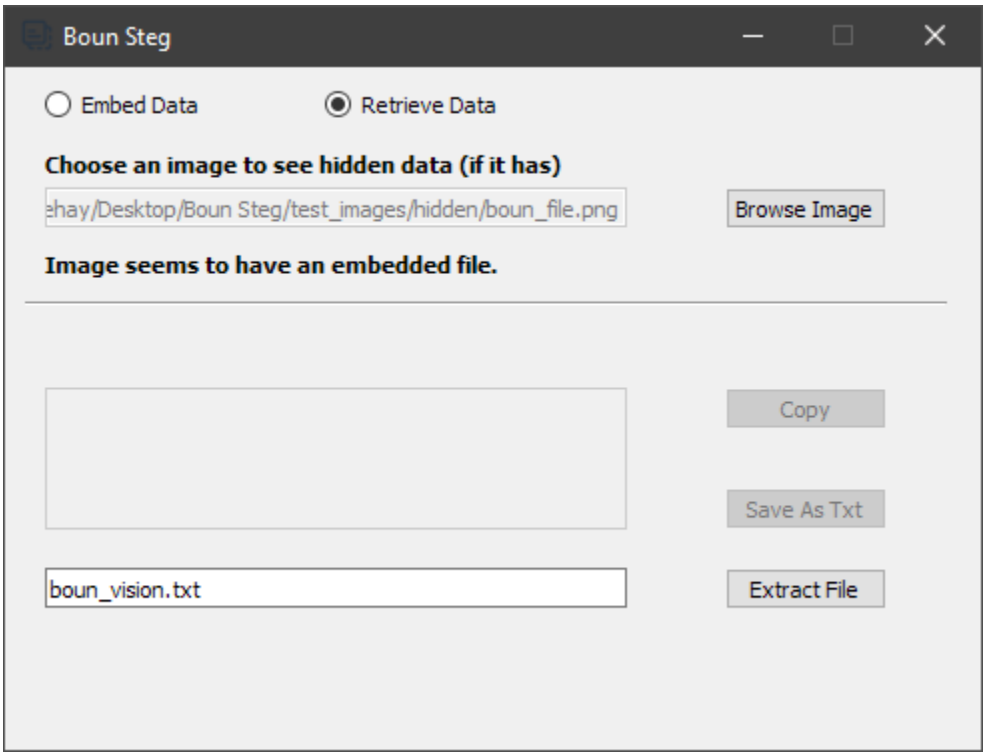

Boun Steg
Embed Data
Retrieve Data
Choose an image to see hidden data (if it has)
y/Desktop/Boun Steg/test_images/hidden/boun_secret.png
Browse Image
Image seems to have a hidden text.
The truth is rarely pure and never simple.
Oscar Wilde
Copy
Save As Txt
Extract File

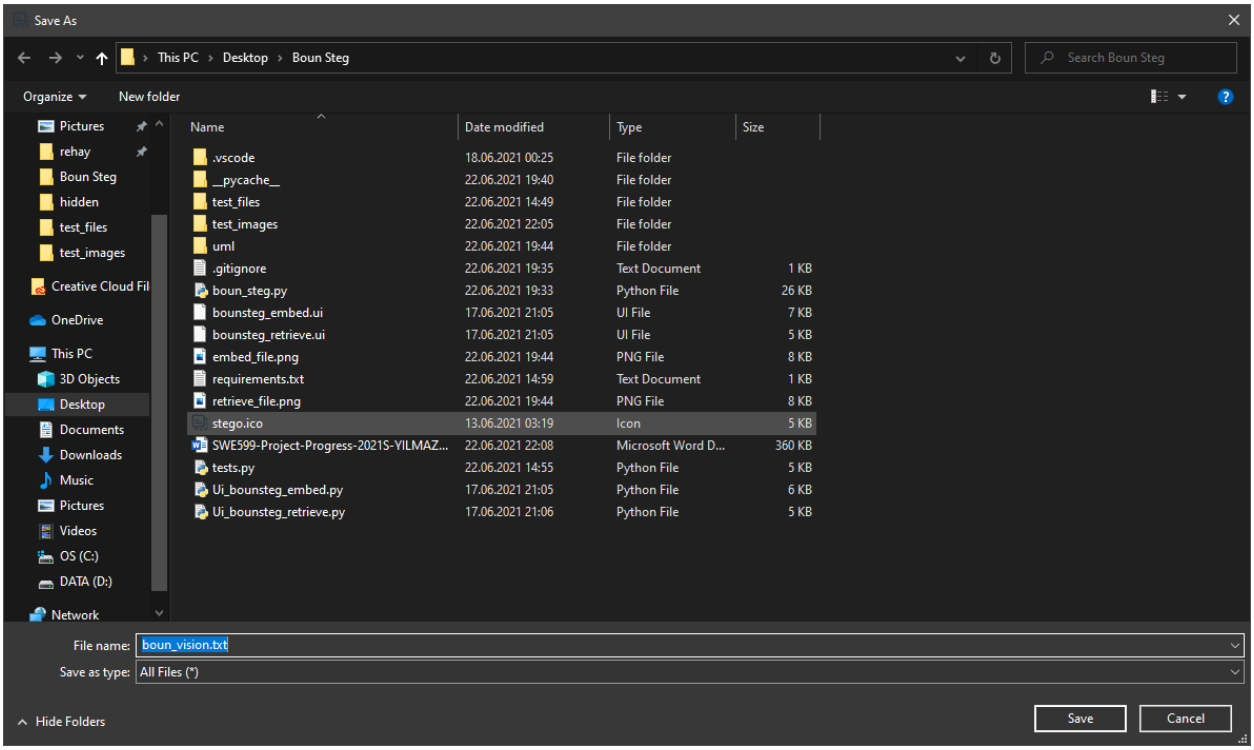
Image without data:

Image with data:

**File Embedding**




Save As
This PC › Desktop › Boun Steg
Search Boun Steg
Organize
New folder
Pictures
rehay
Boun Steg
hidden
test_files
test_images
Creative Cloud Fil
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
OS (C:)
DATA (D:)
Network
Name
Date modified
Type
Size
.vscode 18.06.2021 00:25 File folder
__pycache__ 22.06.2021 19:40 File folder
test_files 22.06.2021 14:49 File folder
test_images 22.06.2021 22:05 File folder
uml 22.06.2021 19:44 File folder
.gitignore 22.06.2021 19:35 Text Document 1 KB
boun_steg.py 22.06.2021 19:33 Python File 26 KB
bounsteg_embed.ui 17.06.2021 21:05 UI File 7 KB
bounsteg_retrieve.ui 17.06.2021 21:05 UI File 5 KB
embed_file.png 22.06.2021 19:44 PNG File 8 KB
requirements.txt 22.06.2021 14:59 Text Document 1 KB
retrieve_file.png 22.06.2021 19:44 PNG File 8 KB
stego.ico 13.06.2021 03:19 Icon 5 KB
SWE599-Project-Progress-2021S-YILMAZ... 22.06.2021 22:08 Microsoft Word D... 360 KB
tests.py 22.06.2021 14:55 Python File 5 KB
Ui_bounsteg_embed.py 17.06.2021 21:05 Python File 6 KB
Ui_bounsteg_retrieve.py 17.06.2021 21:06 Python File 5 KB
File name: boun_vision.txt
Save as type: All Files (*)
Hide Folders
Save
Cancel

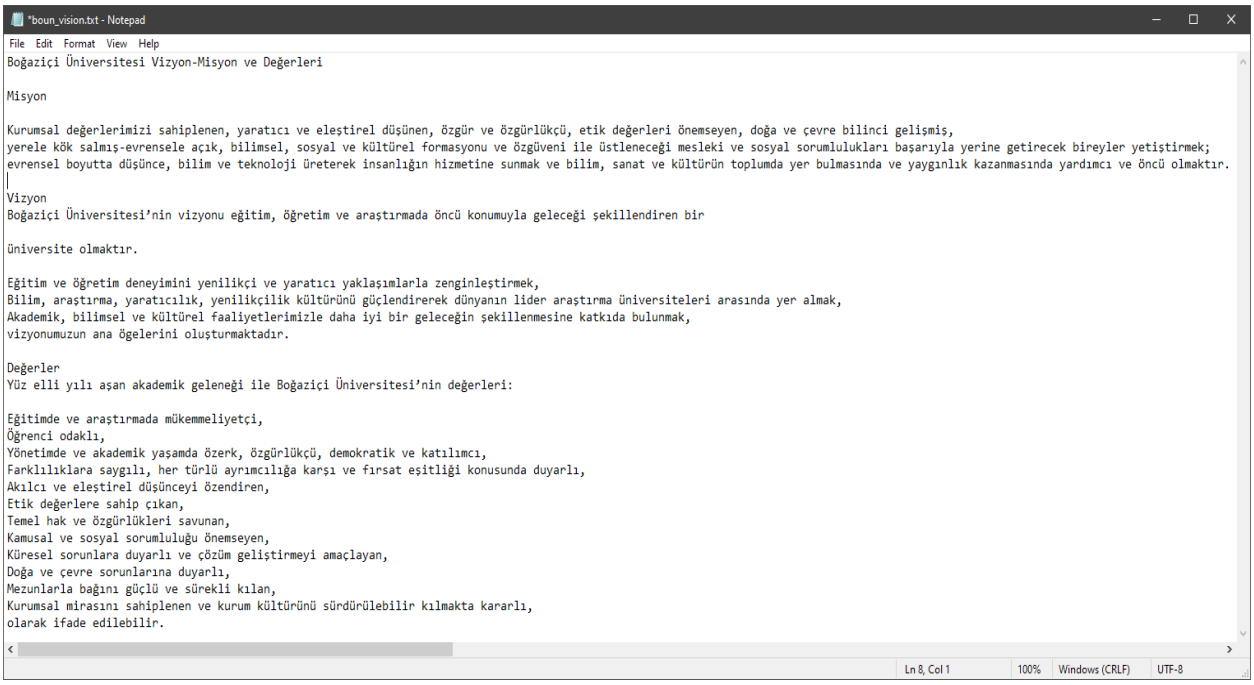

*boun_vision.txt - Notepad
File Edit Format View Help
Boğaziçi Üniversitesi Vizyon-Misyon ve Değerleri
Misyon
Kurumsal değerlerimizi sahiplenen, yaratıcı ve eleştirel düşünen, özgür ve özgürlükçü, etik değerleri önemseyen, doğa ve çevre bilinci gelişmiş,
yerele kök salmış-evrensele açık, bilimsel, sosyal ve kültürel formasyonu ve özgüveni ile üstleneceği mesleki ve sosyal sorumlulukları başarıyla yerine getirecek bireyler yetiştirmek;
evrensel boyutta düşünce, bilim ve teknoloji üreterek insanlığın hizmetine sunmak ve bilim, sanat ve kültürün toplumda yer bulmasında ve yaygınlık kazanmasında yardımcı ve öncü olmaktır.
Vizyon
Boğaziçi Üniversitesi'nin vizyonu eğitim, öğretim ve araştırmada öncü konumuyla geleceği şekillendiren bir
üniversite olmaktır.
Eğitim ve öğretim deneyimini yenilikçi ve yaratıcı yaklaşımlarla zenginleştirmek,
Bilim, araştırma, yaratıcılık, yenilikçilik kültürünü güçlendirerek dünyanın lider araştırma üniversiteleri arasında yer almak,
Akademik, bilimsel ve kültürel faaliyetlerimizle daha iyi bir geleceğin şekillenmesine katkıda bulunmak,
vizyonumuzun ana ögelerini oluşturmaktadır.
Değerler
Yüz elli yılı aşan akademik geleneği ile Boğaziçi Üniversitesi'nin değerleri:
Eğitimde ve araştırmada mükemmeliyetçi,
Öğrenci odaklı,
Yönetimde ve akademik yaşamda özerk, özgürlükçü, demokratik ve katılımcı,
Farklılıklara saygılı, her türlü ayrımcılığa karşı ve fırsat eşitliği konusunda duyarlı,
Akılcı ve eleştirel düşünceyi özendiren,
Etik değerlere sahip çıkan,
Temel hak ve özgürlükleri savunan,
Kamusal ve sosyal sorumluluğu önemseyen,
Küresel sorunlara duyarlı ve çözüm geliştirmeyi amaçlayan,
Doğa ve çevre sorunlarına duyarlı,
Mezunlarla bağını güçlü ve sürekli kılan,
Kurumsal mirasını sahiplenen ve kurum kültürünü sürdürülebilir kılmakta kararlı,
olarak ifade edilebilir.
Ln 8, Col 1
100%
Windows (CRLF)
UTF-8

Image without file:

Image with file:

## Conclusions

Image steganography is a very powerful tool. As seen above it can hide text and all kinds of files inside a simple image. Especially very long texts can be hidden inside of simple image files because of their small sizes. It is not possible to perceive any difference between old and new images by the human eye. You can publish stego image even on the most crowded social sites, yet, so few people would know it has some secret data inside or maybe a handful could retrieve the data. If it is combined with encryption, without knowing the private key it would be almost impossible to retrieve the original data. The downside of image steganography method is that it is easy to detect if the image has secret data embedded with stego analyzing tools.

Future work includes combining some encryption algorithm to save the data encrypted so no other third party could read the clear text message or file. Since it is a python script originally, it can be used in macOS, or Unix-based systems. Application can be made standalone for those operating systems in the future if needed. Different image formats might be added such as JPG.

One of the most challenging things while doing the project was to hide and retrieve the data without losing any non-standard English letters. A solution to that was, encoding the data with Base64 algorithm by using UTF-8 encoding that can encode 1.112.064 characters which cover a big part of letters and symbols of many different languages.

# Cited References

## Some Additional Information

- Used non-standard Python modules: Pillow 8.2.0 (for image processing) and PyQt5 (for GUI) 5.15.4
- Used PyInstaller to convert project into standalone Windows Executable. (Exe file)
- Used Python version: Python 3.8.10
- Used PlantUML to create UML diagrams from Python source code:
- PlantUML is distributed under the GPL license. https://www.plantuml.com/
- Free Icon is taken from: https://freeicons.io/office-and-folders-icons/embed-icon-37743
- GitHub repo: https://github.com/rehayilmazlar/BounSteg

## Bibliography


- N.F. Johnson and S. Jajodia, - Exploring Steganography:Seeing the Unseen (1998)
- Eric Cole - Hiding in Plain Sight: Steganography and the Art of Covert Communication (2003)
- Alaa A. Jabbar Altaay, Shahrin bin Sahib, Mazdak Zamani - An Introduction to Image Steganography Techniques - International Conference on Advanced Computer Science Applications and Technologies (2012)
- Harpreet Kaur, Jyoti Rani - A Survey on different techniques of steganography - CSE Department, GZSCCET Bathinda, Punjab, India (2016)
- Frank Y. Shih - Digital Watermarking and Steganography: Fundamentals and Techniques (2017)
- Mr. Jayesh Surana, Aniruddh Sonsale, Bhavesh Joshi, Deepesh Sharma and Nilesh Choudhary - Steganography Techniques – IJEDR (2017)