

INSTITUTE *of*
TECHNOLOGY

CARLOW

Institiúid Teicneolaíochta Cheatharlach



STEGANOGRAPHIC MESSAGING TOOL
FINAL REPORT



Institute of Technology Carlow
C00241234 – Luke Byrne

Table of Contents

Introduction	2
Description of Submitted Project.....	2
The main motive behind steganography:	2
Project Synopsis:	2
Technologies Utilized:	3
Design Implementation.....	4
Welcome Page:	4
Embedding Page:	5
Embedding Page - Cont'd:.....	6
Extraction Page:	7
Extraction Page - Cont'd:	8
Project Evaluation	9
Accomplishments Achieved:	9
Unsuccessful Functions:.....	9
UX and UI Testing:.....	10
Challenges Encountered during the development stages:	11
Personal Project Reflection.....	12
Learning Outcomes Attained:	12
Self-Reflection:	12
Conclusion.....	12

Introduction

The main objective of this document is to give a detailed overview of my journey through the development process of the Steganographic Messaging Tool project. In this final project report, I will give a brief illustration of the finalized tool in which I have developed, the technologies in which I utilized in order to create my steganography tool and the various functionalities in which my tool provides. Additionally, I will also outline my performance during the development stages of this project, the obstacles in which I encountered throughout the project development as well as the positive and negative lessons in which I have also learned from my personal experience after completion of this project. Finally, to conclude this report I will also discuss my achievements in which I have gained after the completion of this project.

Description of Submitted Project

The main motive behind steganography:

Steganography is the methodology of preventing detection of the existence of sender-receiver communication file from an adversary (third party) by obscuring the presence of secretive data being stored and transmitted across a particular transmission medium such as an image, audio, video, or text file.

There are multiple variations of steganography, but my project is primarily focused on image-based steganography using the least significant bit embedding algorithm. The least significant bit embedding algorithm is used to write and embed a secret message within the least significant bits of the pixels within an image.

The LSB embedding algorithm works by embedding the bits of the user's secret message bit-by-bit within the least significant bits of the images pixel in order to create the least amount of distortion to the image in which is being altered. After the image has been embedded the message can be extracted by using a similar algorithm to the embedding algorithm by reading the least significant bits of each pixel in which contained the message.

Project Synopsis:

The purpose of the Steganographic messaging tool is to conceal the presence of a secret message within a carrier image file. The primary use of the steganographic messaging tool is to conceal the existence of any sender-receiver communication occurring within the image cover file from an adversary (third party) in order to allow a recipient to successfully retrieve and extract a hidden message from an image secretly through the use of a covert channel. The main functionality of the steganography messaging tool is to provide the user with the functionality to embed a secret message

within the least significant bits of an images pixels to extract and retrieve the hidden embedded messages in plaintext from the least significant bits of the image pixels using the LSB embedding and extraction algorithm. Typical uses of steganography are digital watermarking on documents as a proof of ownership or to ensure a secure and confidential method of communication or data storage.

Technologies Utilized:

Operating System:

The particular operating system in which I chose to make my steganography messaging tool available on is Microsoft Windows 10 platform.

Programming Language:

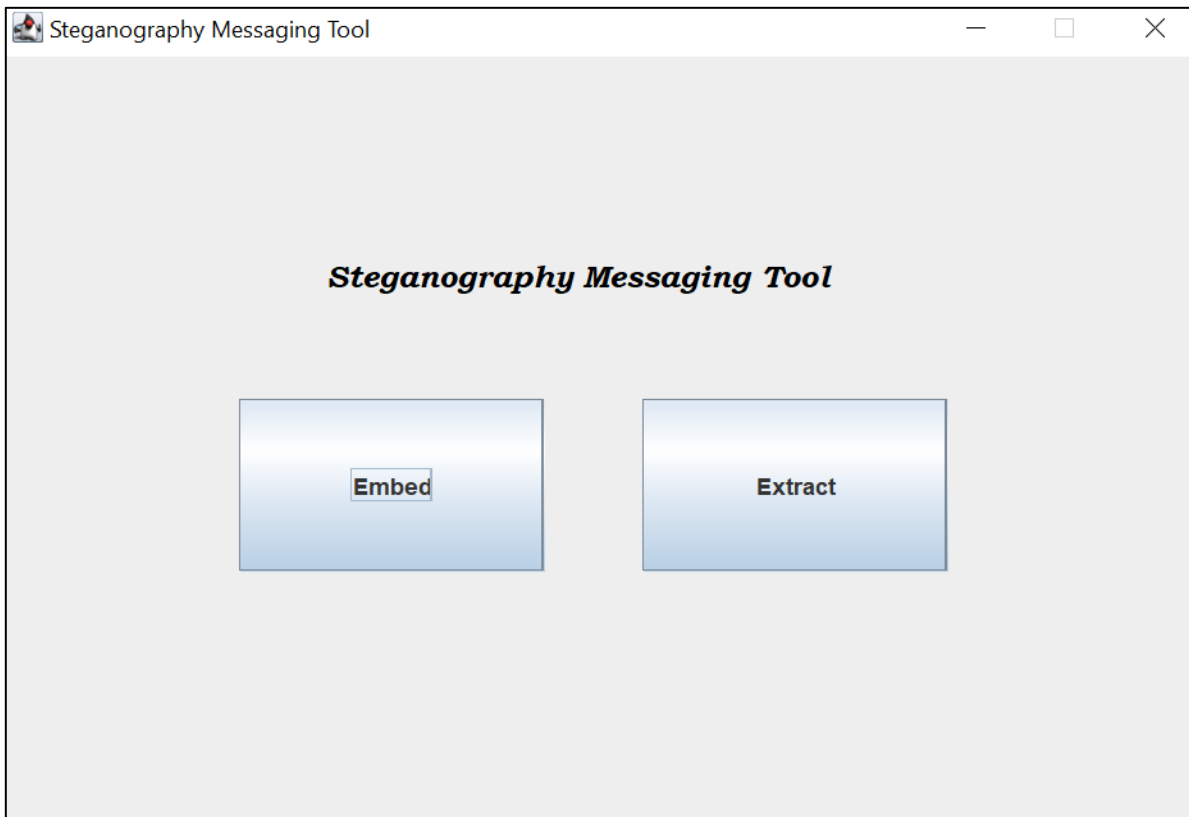
I chose to develop the steganography messaging tool using the Java programming language. The purpose for developing the tool in Java is because Java is the programming language in which I am most familiar with. Additionally, I chose Java to develop my tool as it is a platform-independent language which means the compilation and the execution of java programs can be accomplished on multiple operating systems and environments.

Programming Environment:

For the development of the steganography messaging tools backend I decided to use the Eclipse IDE. I specifically chose to utilize the Eclipse IDE as it includes a built-in frontend graphical user interface (GUI) builder called Windows Builder. Windows builder enables developers to design GUI based applications in a more improved time efficient manner which made my life a little bit easier in comparison to utilizing Java Swing.

Design Implementation

Welcome Page:

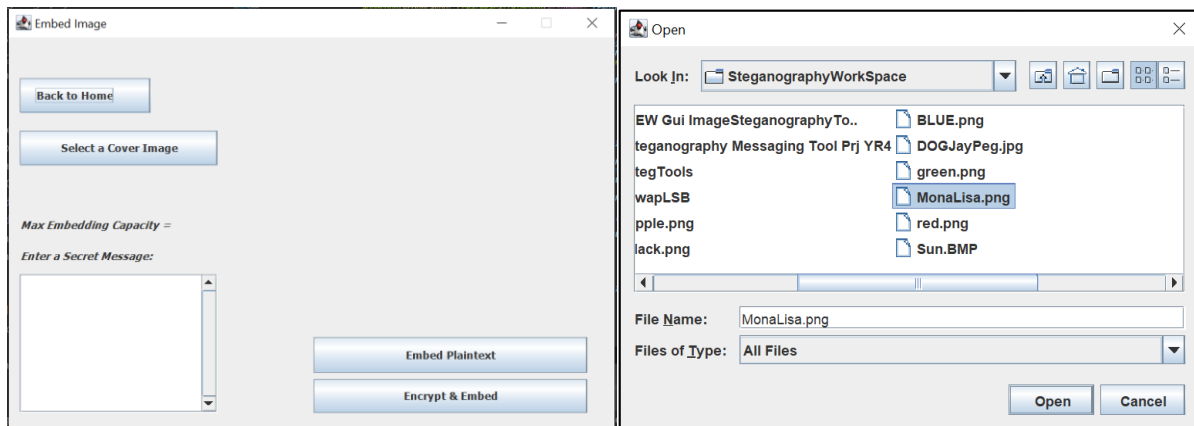


Functionality:

Firstly, when the user clicks on and executes the application they are brought directly to the welcome page. The welcome page presents the end user with a graphical user interface which contains two buttons called "Embed" and "Extract".

The main functionality of the welcome page is to give the user a choice to select whichever function they wish to perform by linking the two buttons to action listeners in order to direct the user either to the embedding page or the extraction page.

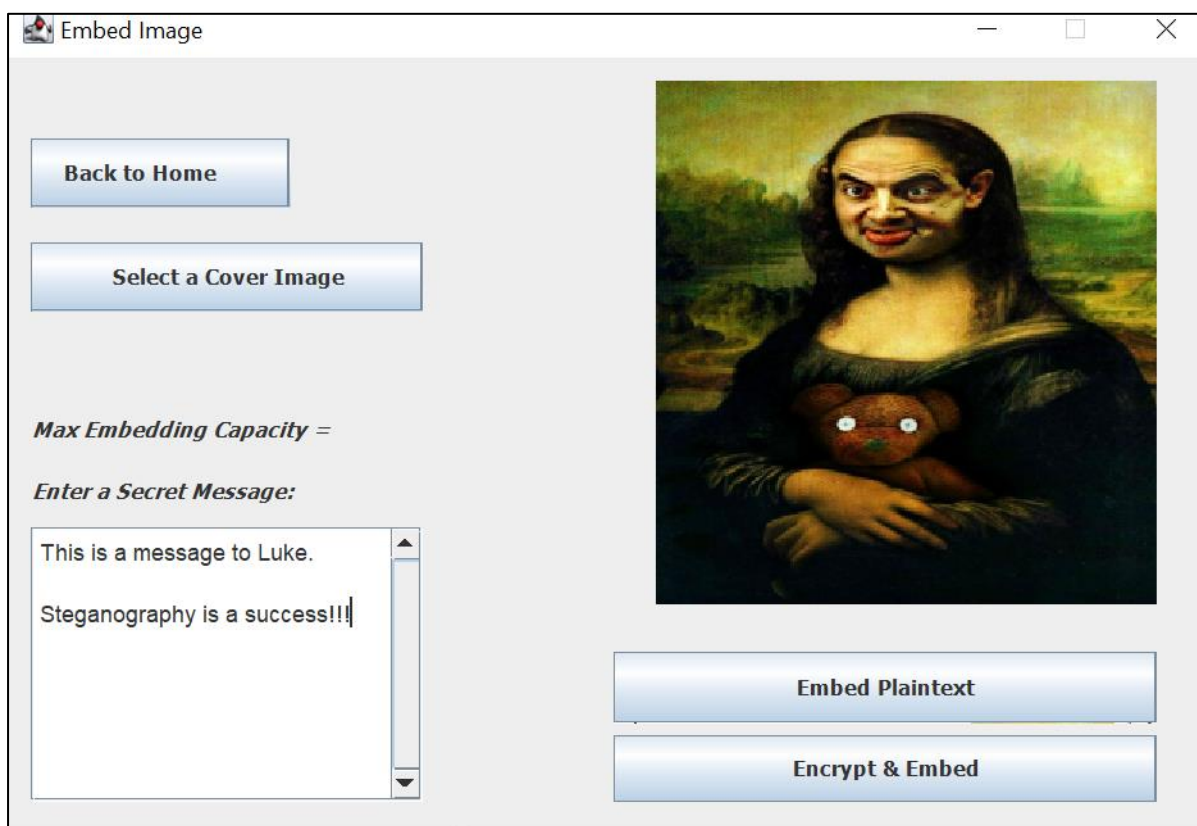
Embedding Page:



Functionality:

In order for the user to be directed to the Embed Image page they will firstly need to click the “Embed” button from the home page. The user is presented with four JButtons and one JEditorPane.

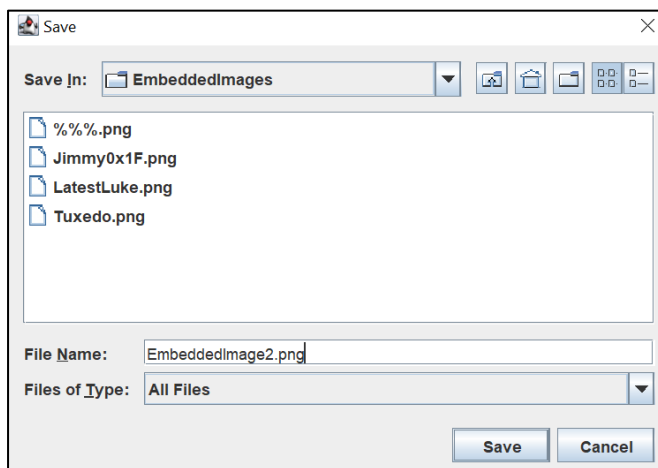
The “Select a Cover Image” JButton is linked to a method using an action listener which allows a user to upload an image using the JFileChooser mechanism. When the user clicks the Select Cover Image button the JFileChooser pops up allowing the user to traverse their directories and allows them to select a carrier image in which they want to use to embed a secret message within.



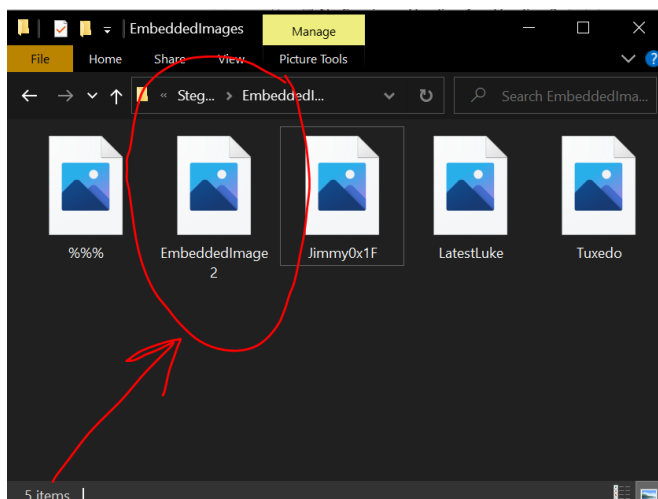
Embedding Page - Cont'd:

After the user has selected an image in which they are using as their carrier file the user will then enter their secret message within the JEditorPane in which is provided to them on the left side of the GUI. When the user is finished entering their secret message withing the text field they will then need to perform the actual embedding function by clicking the “Embed Plaintext” JButton.

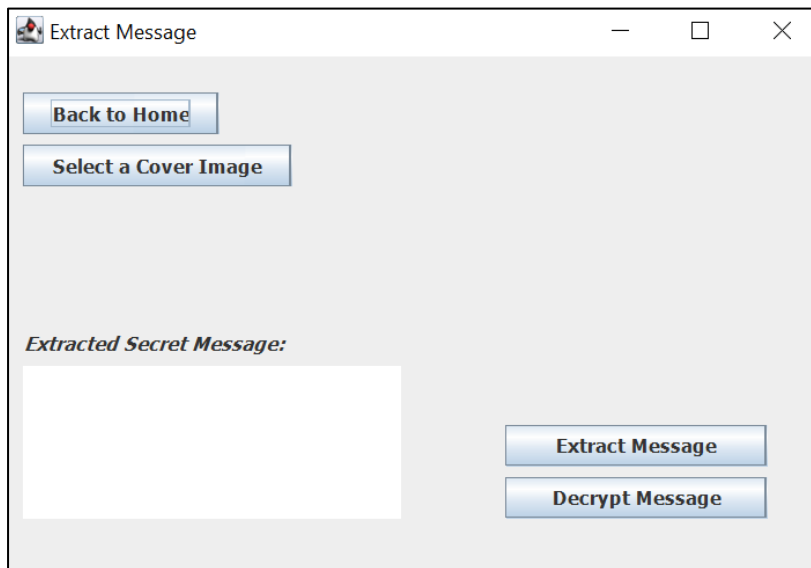
The Embed Plaintext JButton is linked to an action listener which conducts a function that takes the users secret message input within the text field and stores it within byte array. The embedding algorithm is then performed in order to substitute each bit of the user’s secret message from the byte array created with the redundant east significant bits of the image in which has been chosen as the cover image file. After the message bits has been substituted with LSB’s of the image a JFileChooser box will appear prompting the user to save the newly embedded steganographic image in which they have created.



After the user has given the newly created steganographic image a name and a file extension they will need to click the “Save” button in order to write the steganographic image to a directory path of their choice.



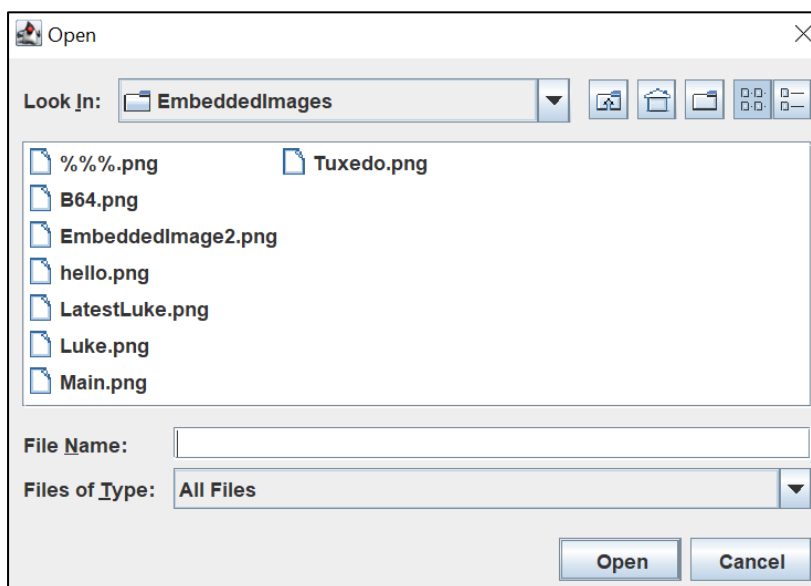
Extraction Page:



Functionality:

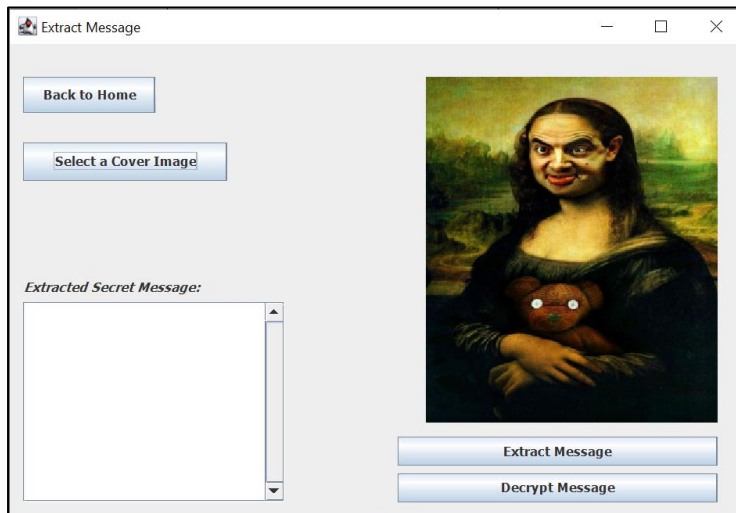
The user can access the extract message page by clicking the "Extract" button on the applications home page.

The "Select a Cover Image" JButton is linked to a method using an action listener which allows a user to choose an image using the JFileChooser mechanism. When the user clicks the Select Cover Image button, the JFileChooser pops up allowing the user to traverse their directories allowing the user to select a steganographic image in which they want to use extract a secret message from.



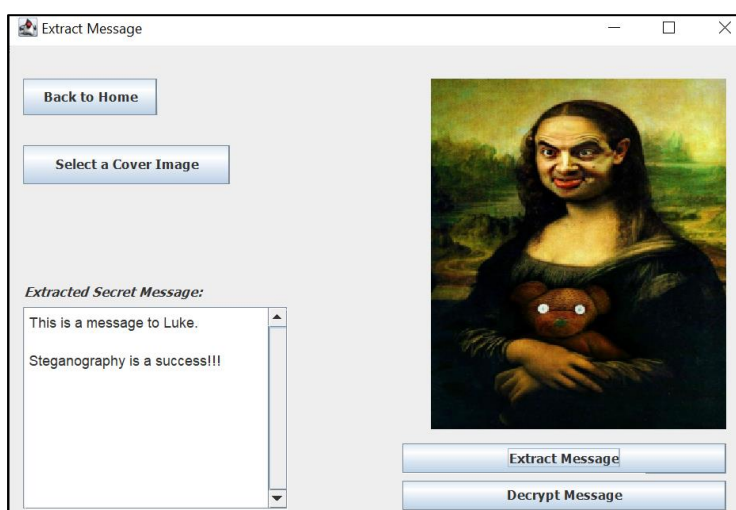
Extraction Page - Cont'd:

After the user has traversed their directories and chosen the cover image in which contains a secret message the user will have to click the open button which will close the file chooser dialogue and it will open the image in which they have selected within a panel positioned on the right-hand side of the GUI.



After the users selected image file has been loaded up within the GUI, the user can now extract the secret message from the least significant bits of each pixel within that image. In order for the user to extract the secret message from the image they will need to click the "Extract Message" JButton. After the "Extract Message" JButton has been clicked, a method that is linked to an action listener will perform the least significant bit extraction algorithm by reading the least significant bits of each pixel in the embedded steganographic cover image.

After the LSB extraction algorithm has been performed the secret message will be extracted from the image and the resulting output will be printed out into the text field box on the left-hand side of the GUI.



Project Evaluation

Accomplishments Achieved:

Title of Successful Functionality		✓ / ✗
1.	Fully functionable Graphical User Interface (GUI)	✓
2.	Users can select and upload an image in which the message will be embedded into.	✓
3.	Uses can enter a secret message in which will be embedded into the cover image.	✓
4.	Plaintext of the secret message can be embedded within a new steganographic image.	✓
5.	The user can save the new steganographic image produced.	✓
6.	The User can read the image in which they embedded a message within.	✓
7.	The plaintext of the message can be extracted from the cover image.	✓

Achieved Functionalities:

The above functionalities were the general core functionalities in which I managed to successfully implement into my steganography project implementation. Each of these functions were assessed manually multiple times in order to be proven to be successfully implemented.

Unsuccessful Functions:

Title of Unsuccessful Functionality		✓ / ✗
1.	The encrypted secret message can be embedded into the cover image.	✗
2.	The ciphertext of the message can be decrypted using the key with which it was encrypted.	✗

Functionalities Not Achieved:

Unfortunately, I could not manage to implement the above functionalities into my project due to time constraints and as well as that I have spent too much time on the other core functionalities. I feel that my lack of proficiency with programming has slowed me down throughout this project implementation, but I am going to further work on this project even after the deadline to implement the remaining functionalities in which I unfortunately could not circumvent. Overall, the encryption of the message would have been nice to have implemented, the project is still a success as it functions correctly to embed plaintext within images.

UX and UI Testing:

My main goal before developing this tool was to create the graphical user interface as simple and effective as possible in order to provide end users the functionalities to perform basic image steganography embedding and extraction. To ensure that my application is working and does exactly what it is supposed to do I conducted multiple manual usability test cases.

Usability Test Cases:

1. The first test case involved me using the application by assessing all the functionalities and buttons as well as observing the speed of the application and how easy it is for me to conduct steganography.
2. The second test case involved my father using the application to assess his performance whilst using the application to conduct image steganography embedding and extraction.
3. The last test case involved my girlfriend using my application. I got my girlfriend to perform basic image steganography by embedding a secret message within an image then extracting that message using the extraction button from the image produced.

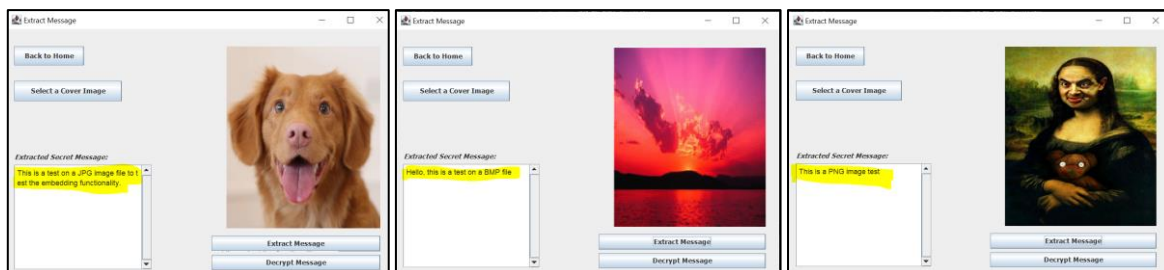
Findings:

As a result of performing the above three usability test cases scenarios it is clear that the usability of my tool is extremely simplistic as all three test cases were considered a success as steganography was able to be performed by all three test case users.

File format testing:

Another test in which I performed was to embed and extract a message using various image file formats such as PNG, JPG, and BMP. All three formats worked accordingly therefore the tests were successful as the message could be retrieved as a result.

Findings:

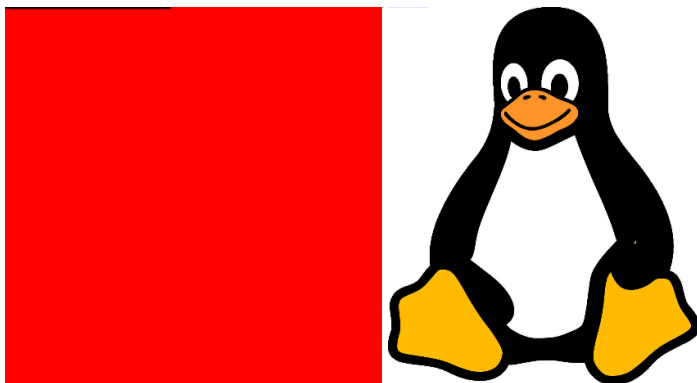


Challenges Encountered during the development stages:

During the development stage of this project, I have certainly encountered many different issues from a programming perspective. The most challenging aspect of this project was developing the code for the least significant bit (LSB) embedding procedure. The reason I found the embedding process to be the most challenging aspect of this project was because there was very little information online on forums or in research papers with regards performing image steganography through the user of integer arrays [].


Initially when I developed the LSB embedding method the message was being embedded incorrectly into the image causing a lot of distortion to the pixels within the image. The algorithm in which I initially used would only embed the bits of the message into the most significant bits of the blue components of each pixel in the (RGB) image causing disruptions to the physical appearance of the images in which I was using.

Here is the replicated issue in which I was encountering when the LSB embedding algorithm was performing incorrectly. The embedding algorithm was performing the substitution of the message with the most significant bits (MSB) of the image hence creating distortion in the blue components of the image. The distortion can be seen in both images below.



Another challenge in which I encountered throughout the development phase of this project was determining the length of my embedded message when extracting it from the image. After a lot of research online I produced the solution to append two special characters (~*) to the end of user's secret message when embedding occurs. The purpose of appending these special characters onto the end of the user's secret message is to determine the termination point of the embedded string to know how much data to extract from the image when performing the LSB extraction method to retrieve the hidden message. Here is a demonstration of the output from the console below.

```
Image Successfully Embedded !!!!  
The Secret Message being embedded is --> Hello, this is a secret message from Luke :)~*
```



Personal Project Reflection

Learning Outcomes Attained:

As a result of me taking on the responsibility of this project I have certainly developed many skills and techniques in order to complete a project successfully whilst sticking to the scope and on time for the deadline. Before starting this project, I was very unsure about the development side of things within this project as I am not the most proficient programmer in the world therefore this project has given me a greater insight to the Java programming language which has certainly made me a more confident programmer further helping me to enjoy programming more now. Over the course of the year the project has helped me to become a more punctual for deadlines and I have gained a lot of valuable knowledge from reading reports and drafting my own research document on the topic of steganography.

Self-Reflection:

If I were to perform this project again from start to finish, I would start read more research papers on LSB embedding before attempting to tackle the embedding algorithm. In addition to this, I would choose to read my image into a Byte array instead of an integer array [] as it is harder to embed into every 8th bit because integer arrays are represented in 32-bits (4 Bytes). From my experience of researching how to embed into integer arrays I discovered that there seems to be a lot more information on forums and papers utilizing byte arrays [] for steganographic embedding.

In my spare time I am going to keep up with programming as I am starting to enjoy it a lot more than I did at the beginning of this project. When I complete 4th year, I am going to repeat this project in my own time utilizing the Python programming language as there is a lot of example projects and libraries online which use Python for image steganography that I would like to try out.

Conclusion

The initial project brief outlined that the objective of this project is to conceal the presence of a secret message embedded within a transmission medium of my choice (Image, Audio, Video, etc..) and to provide the functionality to extract the secret message from the chosen cover medium without adversely affecting the fidelity of the transmission medium.

To briefly conclude, I consider this project to be a success overall as it is truly clear from the test case samples in which I have conducted have determined that the project implementation is extremely simplistic, effective, and fully functional in order to perform the core practices of image steganography across multiple image file formats.