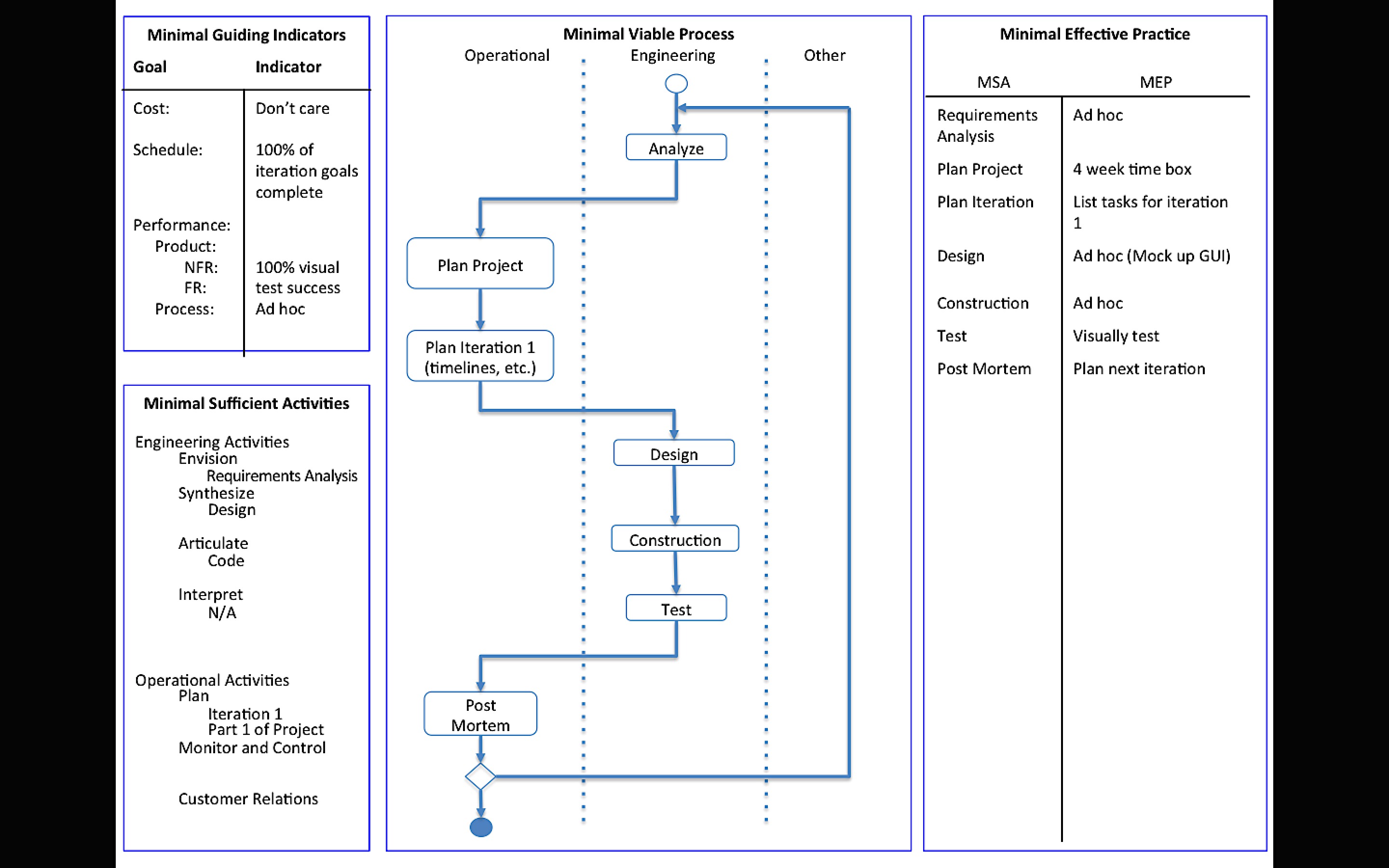
Iteration 1

# Process for Iteration 1



# Requirements Analysis

Given a secret image file and two innocent image files, the tool should

* Be able to read in image files and store the pixel information
* Use the extended visual cryptography scheme to encode the secret image pixels into the two innocent images
* Store the encoded images in new image files
  + The filenames and location can be specified by the user. If not, the files are named share1 and share2 and gets stored on the Desktop.

Given two encoded image files, the tool should

* Be able to read in the files and store the pixel information
* Use the extended visual cryptography scheme to decode the secret image from the encoded images (similar to super imposing them)
* The image revealing the secret gets stored in a new image file
  + The filename and location can be specified by the user. If not, the file is named secretMsg and gets stored on the Desktop.

The visual cryptography tool will only work with PNG and JPEG images. The images involved with the encoding process must have the same dimensions. The tool can handle images of any coloring.

# Plans for Project

Iteration 1:

* Create a graphical user interface
* Get the tool working for strictly black and white images
* Test the tool to check the quality of the encoded shares and the decoded message

Iteration 2:

* Research and implement the visual cryptography scheme with gray scale images
* Add features to project to help boost robustness (i.e. add in checks to keep the user from breaking the tool easily)

Iteration 3:

* Add the ability to encode and decode multicolor images

Iteration 4:

* Analyze the tool and look for ways to improve efficiency (performance and memory storage)

# Plans for Iteration 1

* Clean up the code developed over Summer 2015 and keep only the material relevant to this project.
* Add a graphic user interface to increase the quality of the project.
* Analyze the results of current black and white extended visual cryptography scheme to look for ways of improvement or prepare for it to evolve into handling gray scale images.

# Design

The design portion of Iteration 1 was focused on designing a graphical user interface for the visual cryptography tool. While thinking about how I wanted the user to work with the tool, I decided that three designs were needed: main/welcome page, encode page, and decode page. Then, I created mockups on engineering paper to plan the elements to be placed in the java frames. Figure 1 shows the mockups and Figure 2 displays the current user interface for the visual cryptography tool.

|  |
| --- |
|  |
| Figure 1: Design of Main/Welcome Page (left), Design of Encode Page (middle), Design of Decode Page (right) |

|  |
| --- |
|  |
| Figure 2: Main/Welcome GUI (top), Encode GUI (center), Decode GUI (bottom) |

# Construction

The construction for Iteration 1 involved creating a graphical user interface and merging my previous developed code with the new interface. In order to better design the GUI, I used the NetBeans IDE 8.0.2. This IDE allows you to drag and drop items from the javax.swing Library into your frame and more easily align objects. Once all those elements were functioning, I added in the classes necessary for the encoding and decoding processes. See source code at the end of documentation.

# Test

Due to time constraints, I was only able to perform one encoding test and one decoding test. Figure 3 shows the PNG image files I used for encoding.

|  |
| --- |
|  |
| Figure 3: Secret Image to be Encoded (left), Innocent Image 1 (middle), Innocent Image 2 (right) |

The input for encoding can be seen in Figure 4, and it shows how the graphical user interface works with the encoding process.

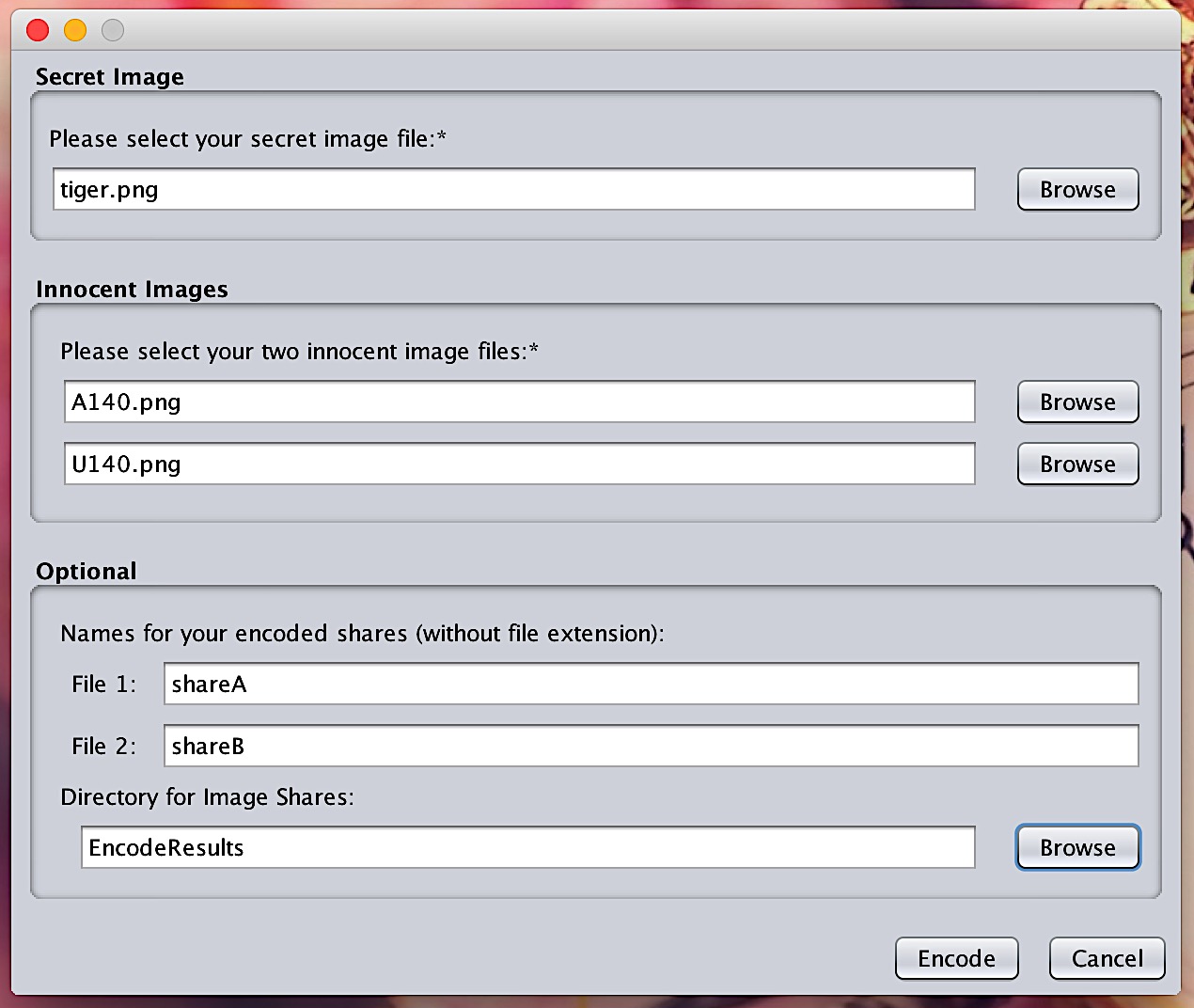


Figure 4: GUI before encoding the images in Figure 3.

Figure 5 displays the results of the encoding process. These images were used as the inputs for testing the decoding process as well.

|  |
| --- |
|  |
| Figure 5: Encoded Share A (left), Encoded Share B (right) |

The input for encoding can be seen in Figure 6, and it shows how the graphical user interface works with the decoding process.

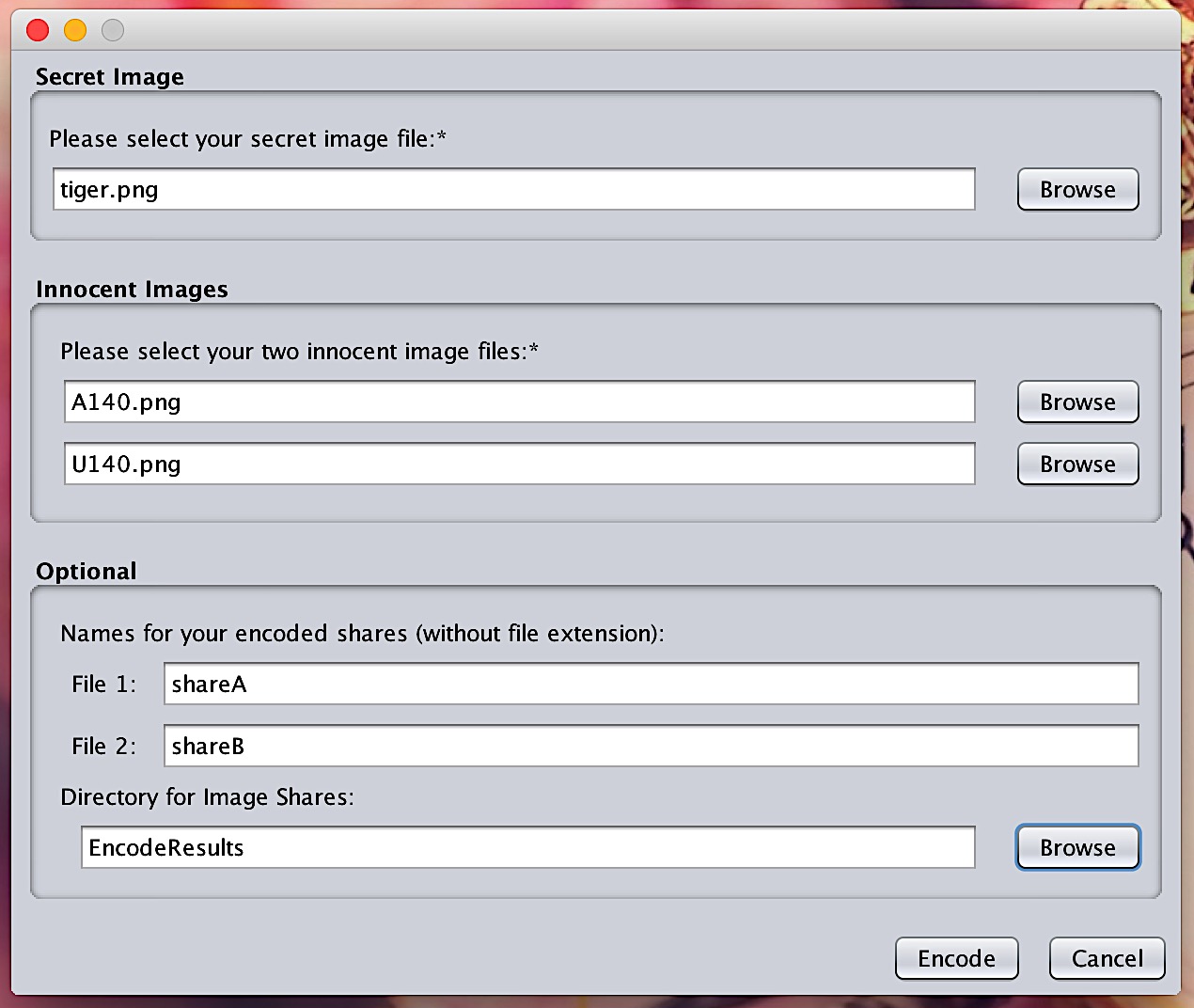


Figure 6: GUI before decoding the images in Figure 5.

The result of the decoding Share A and Share B can be viewed in Figure 7.

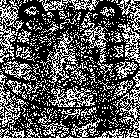


Figure 7: Result of decoding shares A and B from Figure 5.

# Post Mortem

During Iteration 1, I created a graphical user interface to allow for better interaction with the visual cryptography tool. The user interface lets you visual find the image files you wish to use for encoding or decoding without worrying about typing out the absolute path to the file. Once the basics for the GUI were completed, I added some of the classes I made over summer into the project. Currently, the project can encode and decode images that contain only black or white pixels. Thus, the first two tasks listed under the Iteration 1 plan were met.

I did not have a significant amount of time for testing during this iteration. Hence, I only performed one encoding test and one decoding test. The results can be seen in the Testing section.

There is an issue with the matrices I am using to encode the black and white images. Share A looks fine, and you can tell the image is the letter A. Share B, on the other hand, is not as clear as it should be. A person should be able to look at Share B and only see the letter U. With this file, you can make out the letter A and see a portion of the tiger’s face. This issue will be investigated during iteration 2 along with adding functionality for the project to handle gray scale images.

As for the decoding test, you can see a tiger face in the decoded image (Figure 7). However, the decoding process adds more noise to the image making it difficult to see all the aspects of the secret message. The algorithm I am using should permit a user to print decoded images onto transparencies and physically stack the images on top of one another to reveal the secret image. More research needs to be done to see if the background noise can be reduced or if the requirements of no technology needed for the decoding needs to be altered.

During Iteration 2, I plan on adding the ability to encode and decode gray scale images and make the GUI more robust. Based on the little amount of time I had to test, the task of applying more robust error checking may be delayed until Iteration 4 since it is not a task necessary for the encryption or decryption of images. The delay will only occur if the gray scale implementation and testing takes longer than expected.

# Source Code

TBD