**Steganography Project Design**

Linden Crandall, Jonathan Mainhart, Zhihua Zheng

University of Maryland Global Campus

CMIS 495: Current Trends and Projects in Computer Science

Prof. Majid Shaalan

April 12, 2022

Table of Contents

[List of Figures 3](#_Toc102295917)

[List of Tables 3](#_Toc102295918)

[Overview 4](#_Toc102295919)

[Project Plan 4](#_Toc102295920)

[Requirements Specification 4](#_Toc102295921)

[User Guide 4](#_Toc102295922)

[Test Plan and Results 4](#_Toc102295923)

[Design and Alternate Designs 5](#_Toc102295924)

[Development History 5](#_Toc102295925)

[Conclusion 5](#_Toc102295926)

[Lesson’s Learned 5](#_Toc102295927)

[Design Strengths 5](#_Toc102295928)

[Design Limitations 6](#_Toc102295929)

[Future Improvements 6](#_Toc102295930)

# List of Figures

[Figure 1. Place holder 5](#_Toc102295570)

# List of Tables

[**Table 1:** Place holder 4](#_Toc102295903)

# Overview

Steganography is the practice of hiding secret information inside of something that is not secret. There are many applications of digital steganography, but one of the most common implementations is the practice of embedding a secret message within an image file. Steganosaurus is an image-based digital steganography application written in the Python language. The app allows users to retireve, decode, read, create, and encode secret messges to and from images chosen by the user. The result of an encoded image is a seemingly exact replica of the orginial image, yet the difference lies within the pixel data of the encoded image where the secret messge is hidden, which is exactly how steganography is supposed to work. The purpose of this software is to allow users to have fun and experience how steganography works in a user-friendly way that anyone can enjoy.

Individual Contributions:

Linden Crandall - File I/O for implemennting open and save filechooser, User’s Guide, Phase II assignment, general documentation and app testing.

Jonathan Mainhart -

Zhihua Zheng – Project management. GUI kv files implementation, GUI interaction implementation(stego.py), unittest implementation(stegoTest.py), manual test execution, and fix bugs. General documentation input, submission of project proposal and specifications, Phase I.

# Project Plan

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.

Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci.

# Requirements Specification

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.

Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci.

Aenean nec lorem. In porttitor. Donec laoreet nonummy augue.

Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy.

# User Guide

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.

# Test Plan and Results

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.

Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci.

**Table 1:**  
Place holder

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Aenean nec lorem. In porttitor. Donec laoreet nonummy augue.

Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy.

Fusce aliquet pede non pede. Suspendisse dapibus lorem pellentesque magna. Integer nulla.

Donec blandit feugiat ligula. Donec hendrerit, felis et imperdiet euismod, purus ipsum pretium metus, in lacinia nulla nisl eget sapien. Donec ut est in lectus consequat consequat.

Etiam eget dui. Aliquam erat volutpat. Sed at lorem in nunc porta tristique.

Proin nec augue. Quisque aliquam tempor magna. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Figure 1*. Place holder*

# Design and Alternate Designs

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.

Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus.

# Development History

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.

Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus.

# Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.

## Lesson’s Learned

There were many obstacles that our group faced and overcame during the conception, planning, documentation, development and completion of our application. One of the most important lessons learned was ensuring that our software was compatible with and able to run natively on each of the three major operating systems without any limitations and/or handicaps. One specific roadblock we ran into in the early stages of development was implementing a filechooser that worked in tandem with our GUI framework, Kivy. The initial filehooser implementation worked well on Windows OS, but not on MacOS. We attempted to to fix this, however we soon realized that the current design was inadequate, so we had to regroup and reconsider our implementation. We ultimately decided to use the Kivy Framework’s built-in filechooser which served us quite nicely and got the job done, although it prevented us from implementing each OS’s default file explorer that users would be most used to seeing.

Other lesson’s learned were: learning how to collaborate and develop on a single repository using git, learning the Python language and the Kivy framework and how they work together, and how to have open, honest, and frequent communication among group members so that we are always on schedule and always “in the know” which helped pave the way to the successful completion of this project.

## Design Strengths

Video provides a powerful way to help you prove your point. When you click Online Video, you can paste in the embed code for the video you want to add. You can also type a keyword to search online for the video that best fits your document.

## Design Limitations

Design Limitations for this project include the program performance check, incomplete automated testing implementation, and deprecation warnings for a few methods used in the unit test. Considering the 8-week time limitation and the small-scale of the project, the performance check, such as the time complexity and the space complexity, were not arranged throughout the development. Based on the size of the image, there is an apparent time delay to execute the Save Image function. For instance, an image size of 24MB requires about 13 seconds of execution time. Therefore, the current program will be more suitable to work with the smaller size images.

(Below are taken from Jonathan’s Phase 3 doc – Problems Encountered and Reevaluation of the Decisions.)

Next, we are unable to implement automated testing for some parts of the application due to some design choices. Notably, exception testing cannot be automated for methods which handle their own exceptions instead of passing the exception to the caller. A major refactor would be required to fix this issue which may affect our ability to deliver on time. We decided to press forward with manual testing the exceptions.

Finally, another small issue with the testing framework came up while testing on Windows systems. The framework presented depreciation warnings for a few methods used in the tests themselves. While not something that needs to be corrected right now, this is something that should be corrected before Python 3.12 is released.

## Future Improvements

Reevaluating and improving the program performance will be the main focus for future improvements. Such as reviewing the code and choosing a better-suited programming algorithm to reduce the program execution time. Furthermore, the insufficient research on the Kivy language and its compatibility with Tkinter caused the rollback from the implementation phase to the previous project design phase. Hence, sufficient research on the limitation of the used languages and algorithms will be stressed during the project design phase.