Steganography Tool for Image/File Hiding

### 1. Introduction

This report details a desktop application created to perform digital steganography. The primary objective of this project is to provide a simple, user-friendly tool that can hide a secret text message inside a standard image file and, conversely, extract a hidden message from such an image.

The tool features a graphical user interface (GUI) that allows a user to select an image, type a message, and embed it. They can also use the same interface to open a modified image and extract the hidden data. This project serves as a practical introduction to the concepts of steganography and data manipulation at the byte level.

**2. Core Concept: What is Steganography?**

Steganography is the art and science of hiding information within other, non-secret files or messages. The word itself means "covered writing."

Unlike **cryptography**, which scrambles a message to make it unreadable (encryption), **steganography** hides the very *existence* of the message. The goal is for a third party to look at the carrier file (in our case, an image) and have no suspicion that it contains a secret.

This tool uses a common technique called **Least Significant Bit (LSB)** insertion. Here’s a simple breakdown:

* Every pixel in an image is made of color values (Red, Green, Blue).
* Each color value is a number (e.g., 255). In binary, this is 11111111.
* The "Least Significant Bit" is the very last bit (the '1' on the far right).
* Changing this last bit (from 11111111 to 11111110) causes a change in color so tiny that the human eye cannot see the difference.
* By strategically changing the LSBs of many pixels, we can embed the binary data of our secret message into the image, invisibly.

**3. Technologies Used**

This tool was built entirely in Python, using a combination of standard and third-party libraries:

* **Python:** The core programming language for all logic.
* **Tkinter:** Python's built-in standard library for creating the graphical user interface (GUI). It provides the windows, buttons, text boxes, and file dialogs.
* **Pillow (PIL):** The Python Imaging Library, used to open, process, and save image files. It acts as the bridge between our image files and the stepic library.
* **Stepic:** A powerful, simple-to-use Python library that handles the complex LSB steganography logic. It performs the core functions of encoding (embedding) our message into the image and decoding (extracting) it.

**4. User Guide (How to Use the Tool)**

The application has a straightforward interface with two primary functions.

**A. How to Embed (Hide) a Message:**

1. **Run the Script:** Start the program by running the steganography\_tool.py script.
2. **Select Image:** Click the **"Select..."** button. A file dialog will open. Choose a "cover" image to hide your message in. (Note: **PNG** and **BMP** formats work best as they are "lossless" and won't corrupt the hidden data).
3. **Enter Message:** Type the secret message you wish to hide into the large text box.
4. **Embed:** Click the **"Embed Message"** button.
5. **Save New Image:** A "Save As" dialog will appear. Choose a name and location for your new image. **It is crucial to save this file as a .png** to ensure the hidden data is preserved.
6. A "Success" pop-up will confirm the message is hidden and saved.

**B. How to Extract (Reveal) a Message:**

1. **Run the Script:** Open the tool.
2. **Select Image:** Click the **"Select..."** button and choose the image that you suspect contains a hidden message (e.g., the \_hidden.png file you saved earlier).
3. **Extract:** Click the **"Extract Message"** button.
4. **View Message:** If a hidden message is found, the text box will be cleared, and the secret message will be displayed inside it.
5. If no message is found, or if the file is not a valid steganographic image, an error message will be displayed.

**5. Conclusion & Future Improvements**

This project successfully achieved its goal of creating a functional, easy-to-use steganography tool. It demonstrates the core principles of LSB steganography in a practical application.

This tool also serves as a strong foundation for several potential improvements:

* **File Hiding:** The logic could be extended to embed entire files (like a .txt or .zip file) instead of just text.
* **Encryption:** The secret message could be encrypted with a password *before* being embedded. This would add a second layer of security, so even if someone extracted the message, they couldn't read it without the password.
* **Support for More Formats:** While PNG is ideal, research could be done to add support for other file types.
* **Drag-and-Drop:** The GUI could be improved to allow users to drag an image file directly onto the window instead of using the "Select..." button.