Steganography Project Report

Project Title:

Image Steganography GUI Tool Using Python

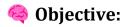


Developed By:

Mahavir Harijan



16 July 2025



The main objective of this project is to build a simple, beginner-friendly GUI tool that allows users to hide and extract secret messages within image files using steganography techniques. It uses Python and provides a clean interface that supports drag-and-drop, file browsing, and works with various image formats.

***** Technologies Used:

- Python 3
- Tkinter (for GUI)
- tkinterDnD2 (for drag-and-drop)
- Pillow (for image handling)
- Stepic (for LSB steganography)

🥓 How It Works:

Hiding Process:

- 1. User selects "Hide Message" mode.
- 2. Uploads image using Drag & Drop or Browse button.
- 3. Enters the message to hide.
- 4. The tool uses stepic to encode the message into image pixels using LSB (Least Significant Bit) technique.
- 5. The output image is automatically saved in the Downloads folder with a name like original name hidden.png.

Extracting Process:

- 1. User selects "Extract Message" mode.
- 2. Uploads an image containing hidden data.
- 3. The tool decodes the embedded message and shows it in a popup.

Supported Image Formats:

- JPG/JPEG
- BMP

(Note: PNG and BMP are recommended for lossless quality)

Project Structure:

Stignography/

├── gui_steganography.py # Main GUI application ├── README.md # Project documentation

User Interface Overview:

- Radio Buttons to choose mode (Hide / Extract)
- Drag & Drop area for image input
- Browse Button as alternative to drag & drop
- Text Entry Field (only visible in Hide mode)
- Action Button: performs hide or extract operation

Challenges Faced:

1. Drag & Drop Support:

- Problem: Tkinter doesn't support drag-and-drop by default.
- Solution: Used tkinterDnD2 package.
- Difficulty: Some systems needed extra handling for event bindings.

2. Image Format Compatibility:

- Problem: Stepic works best with RGB images.
- Solution: Converted JPG/BMP to RGB in memory before encoding.

3. Output Overwriting:

- Problem: Previous image was getting replaced on every hide operation.
- Solution: Used the original filename to generate unique output: name hidden.png

4. Saving Output Automatically:

- Problem: By default, it saved image in working folder.
- Solution: Used Python's pathlib to save directly to Downloads folder.

5. FileNotFound or Format Errors:

- Solution: Added proper exception handling and user-friendly error messages.

Why These Libraries Were Used:

Library **Purpose**

Tkinter To create the graphical user interface (buttons, labels, input fields)

tkinterDnD2 To enable drag-and-drop support in the GUI

Pillow (PIL) To open, process, and convert image files

Stepic To hide and extract secret messages using LSB (Least Significant Bit) steganography

Library Purpose

os / pathlib To manage file paths, extract file names, and auto-save output to the Downloads folder

Use Cases:

- Basic steganography education and learning
- Sending hidden messages within images
- Demonstrating how data can be embedded visually

Output Example:

Input Image: cat.jpg

Message: "Hello, this is hidden!"

Output Image: Downloads/cat_hidden.png

limitations:

- Not encrypted only hides data using LSB
- Compression can destroy hidden data (avoid sending via WhatsApp, etc.)
- Only works with image formats supported by PIL

XX Conclusion:

This project successfully demonstrates how steganography works by hiding data in image files using simple Python libraries. It provides a clean, working GUI with modern features like drag-and-drop and dynamic output saving. The tool is ideal for learning, experimentation, and awareness of how information can be embedded in media.

A Developer Note:

This project helped in understanding:

- How GUIs are built using Tkinter
- How LSB-based steganography works
- Handling user input and file formats safely
- Writing beginner-friendly and clean Python code