**CYBERSECURITY ASSIGNMENT- 1**

**Report on Steganography project.**

**Name:** Madhuri.

**Roll No**:160123737077

**Class:**IT-2

**Steganography Project**: A technique for securely embedding confidential information within digital media, ensuring covert communication without altering the apparent content**(**Tool explored: OpenSteogo**)**

**Date:** 01/09/2025

**Github Repository: https://github.com/sanethfernando/SecureFileSha Steganography Project – Final Report**

**Project Overview**

The objective was to build a steganography system that allows users to securely hide and retrieve secret messages within images. Security and confidentiality were the main focus, ensuring that messages could be embedded without visibly altering the image and could be decoded reliably.  
The system simulates real-world covert communication and data-hiding requirements seen in industries like digital forensics, secure messaging, and confidential data sharing.

**Technologies & Tools Used**

* Programming Language: Python
* GUI Framework: Tkinter (for user interface)
* Image Processing Library: Pillow (PIL)
* Encryption / Encoding: Least Significant Bit (LSB) technique for secure message embedding
* Version Control: Git & GitHub

**System Architecture**

1. User selects an image through the Tkinter GUI.
2. The secret message is converted to bytes and embedded into the least significant bits of the image’s RGB channels.
3. The modified image is saved as a new file (encoded\_output.png) while preserving the original image’s appearance.
4. During decoding, the system reads the RGB channels, extracts the message bits, and reconstructs the original secret message.

**Security Features**

* LSB Encoding: Ensures that the message is hidden without visibly altering the image.
* Unicode Support: Messages in any language or emojis can be securely embedded and retrieved.
* Delimiter-Based Decoding: Uses a specific delimiter (%%END%%) to detect the end of the message, preventing corruption or gibberish.
* Safe File Handling: Original images remain untouched; only copies are modified for message embedding**.**

**Folder Structure**

**SteganographyProject/**

├── docs/

│ └── read.me # Project documentation

├── input\_images/

│ ├── Cat.jpg # Original image

│ └── madhuri\_demo.txt # Secret message file

├── outputs/

│ └── Cat\_secret.png # Encoded image (after hiding message)

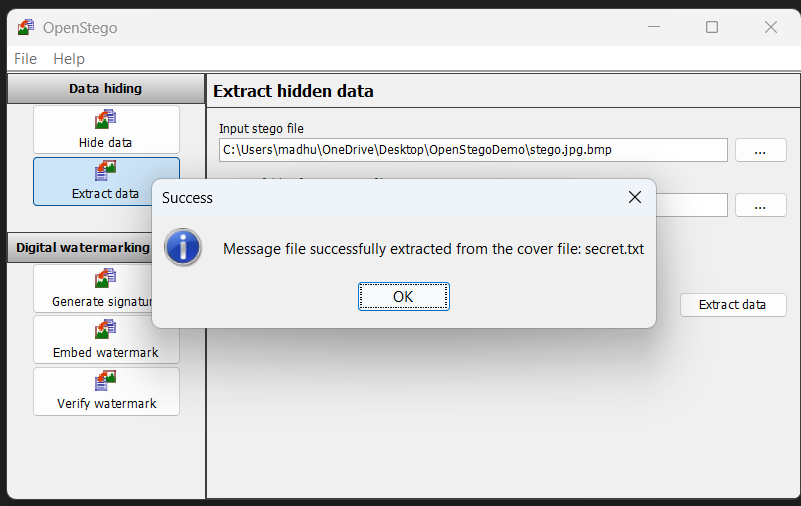
└── src/

├── stego.py # Core encoding/decoding logic

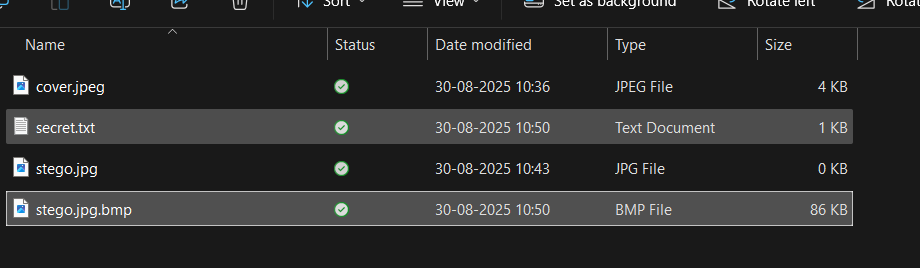
└── stego\_gui.py # Tkinter GUI for user-friendly interface

**Screenshots:**

1.Tool explored: OpenSteogo

****

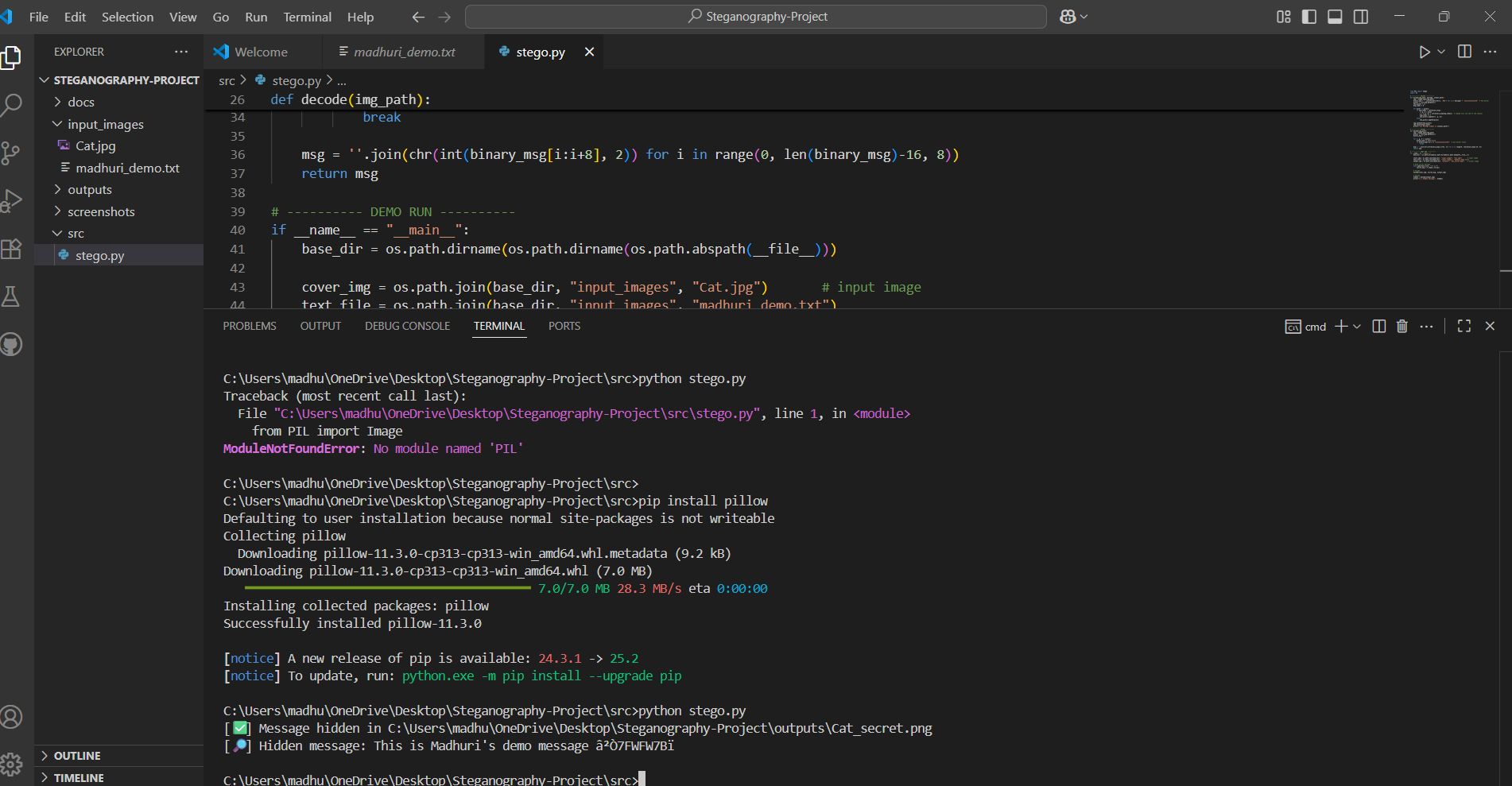
**2. secret Message encoded using tool:**

****

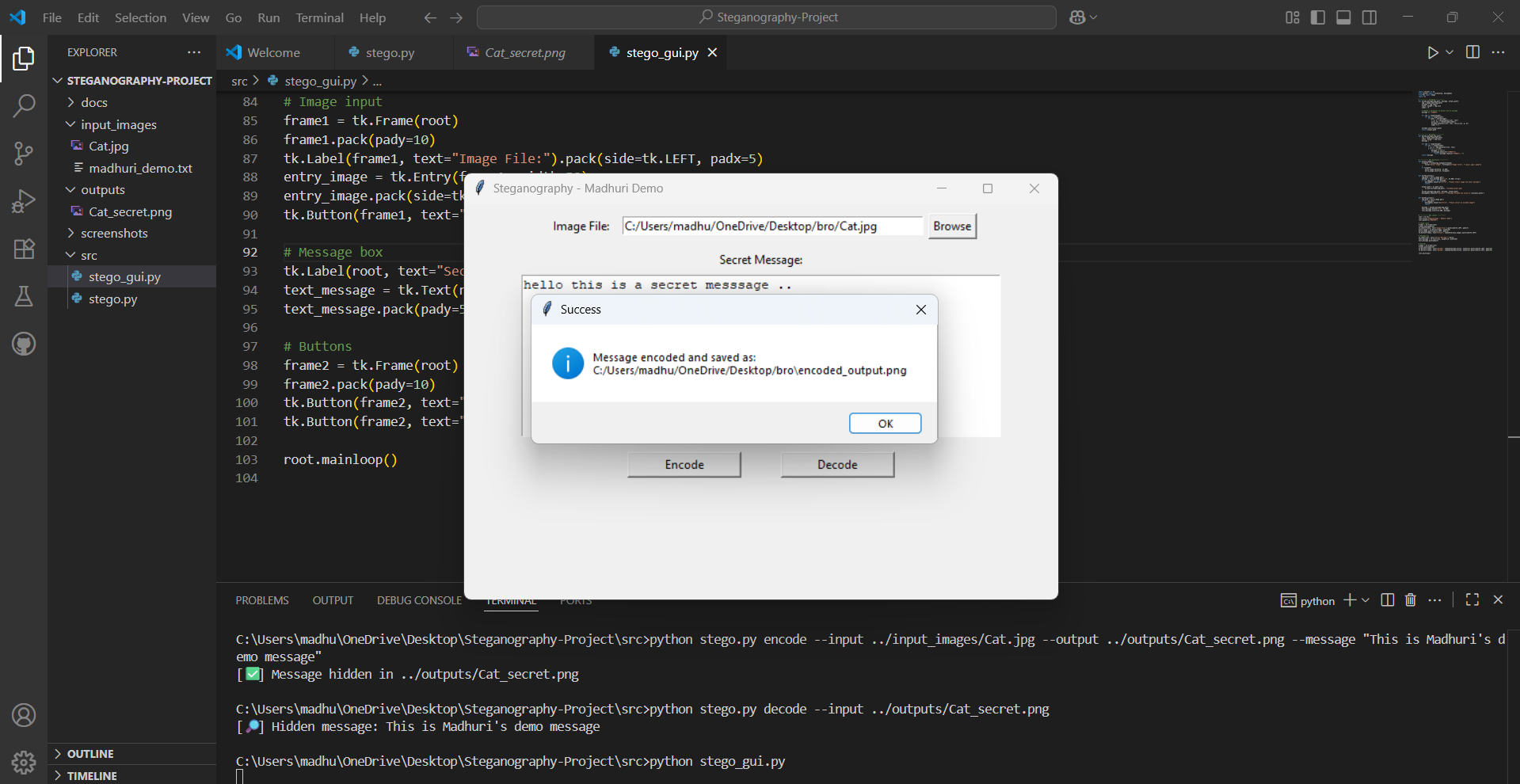
* Cover file → This is the original image you use to hide the message (e.g., Cat.jpg).
* Secret file → This is the message you want to hide (e.g., madhuri\_demo.txt).
* Output file → This is the new image that contains the hidden message (e.g., Cat\_secret.png).

So the output is NOT the secret text itself; it’s the cover image + secret message embedded inside.

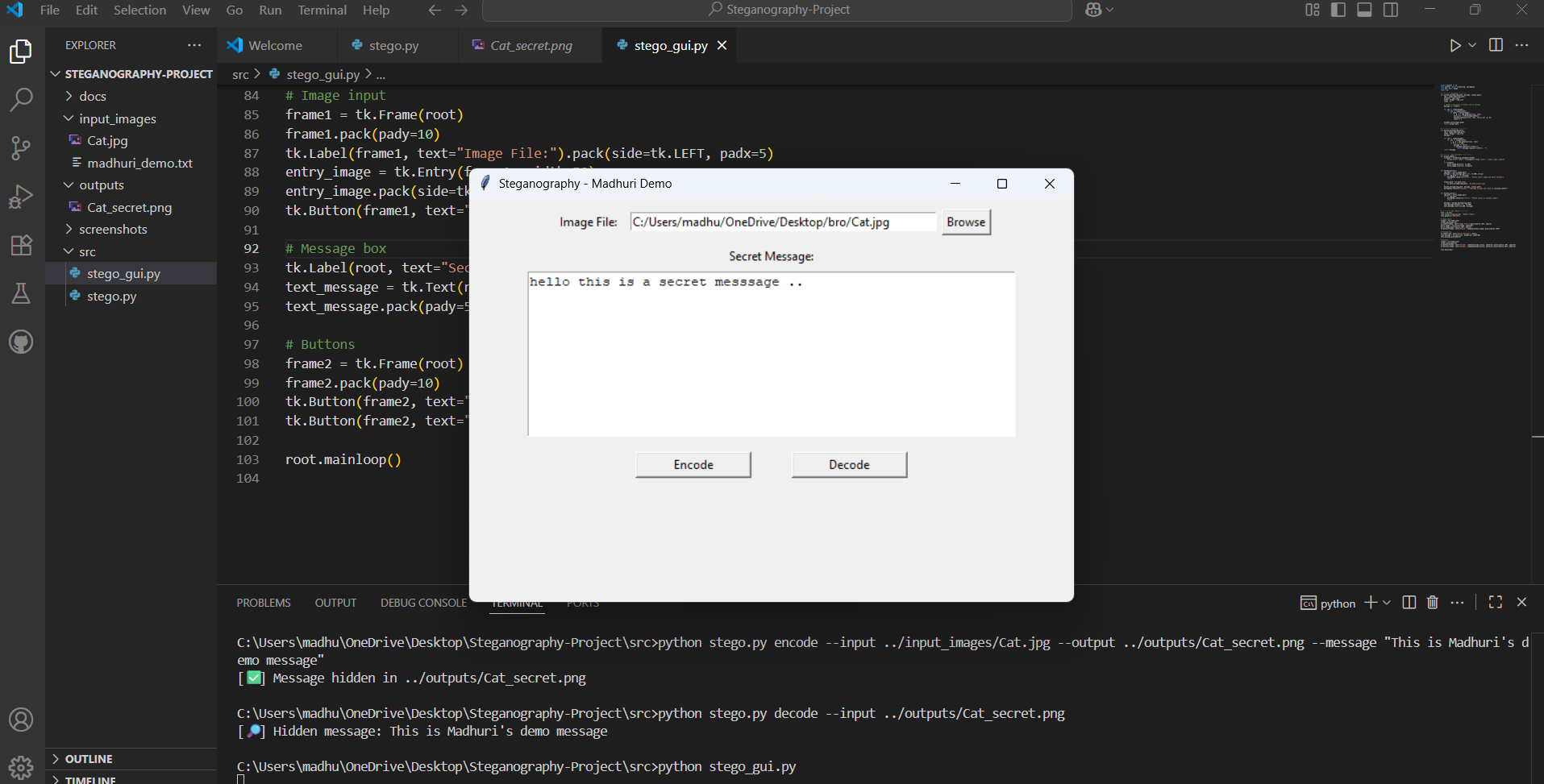
**3.** **OpenStego Interface – Message Encoding**

****

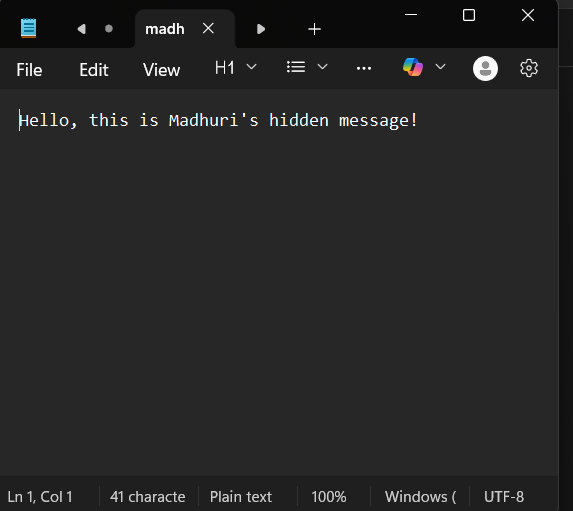
**4.Tkinter GUI – Selecting Image and Entering Message:**

****

1. **Tkinter GUI – Decoded Message Displayed:**



**6.** **Output: Cover.png + Secret.txt**



**Deliverables**

* GitHub Repository with full project code (stego.py, stego\_gui.py) and documentation.
* Sample images demonstrating message encoding and decoding.
* Project Report (this document).

**Learning Outcomes**

* Practical understanding of Python programming for image processing.
* Hands-on implementation of LSB-based steganography.
* Knowledge of secure message embedding and retrieval.
* Experience creating a user-friendly Tkinter GUI.

**Conclusion**

This project demonstrates a functional steganography system for hiding and retrieving messages securely in images. By using LSB encoding, supporting Unicode, and providing a GUI interface, it offers a strong foundation for applications requiring covert communication and secure data handling**.**