**Project Report**

**1. Project Title: Steganography Tool for Image Hiding using Python (CLI & GUI)**

**2. Introduction:**

Steganography is the art of hiding information within other non-secret data to avoid detection. This project involves the development of a **Python-based steganography tool** that allows users to **hide and retrieve secret messages inside image files**. The tool supports both a **Command Line Interface (CLI)** and a **Graphical User Interface (GUI)**, making it accessible for both advanced and casual users.

This project demonstrates how steganography can be used as a means of secure and concealed communication, especially in scenarios where encryption alone might raise suspicion.

**3. Objective:**

The main objectives of the project are:

* To build a steganography tool that allows hiding and extracting text messages from image files.
* To provide both CLI and GUI-based access for better usability and flexibility.
* To explore image processing and data encoding techniques using Python libraries.
* To demonstrate a practical application of cryptography and steganography in cybersecurity.

**4. Technologies & Libraries Used:**

* **Python 3.x**
* **Pillow (PIL):** For image manipulation and pixel-level access.
* **Tkinter:** For creating the GUI.
* **argparse / sys:** For CLI support and argument parsing.
* **OS / base64 (optional):** For file handling and encoding.

**5. Key Features:**

* **Text-to-Image Embedding:** Hide secret messages within PNG image files using LSB (Least Significant Bit) steganography.
* **Message Extraction:** Accurately decode and extract the hidden message from the image.
* **CLI Mode:** Allows advanced users to encode and decode messages through command-line parameters.
* **GUI Mode:** Provides an easy-to-use interface for users to select images and input messages.

**6. Working Mechanism:**

**Encoding:**

1. The user selects a cover image and inputs the secret message.
2. The message is optionally encoded (e.g., base64).
3. The binary data of the message is embedded into the least significant bits of the image pixels.
4. The modified image (stego-image) is saved without any noticeable visual difference.

**Decoding:**

1. The user selects a stego-image.
2. The program scans the image pixels and extracts the LSBs to reconstruct the hidden binary message.
3. The binary is converted back to readable text.

**7. Use Cases:**

* Secure message transfer without detection.
* Educational tool to demonstrate how digital steganography works.
* Enhancing personal privacy for sensitive data.

**8. Learning Outcomes:**

Through building this project, the following concepts and skills were developed:

* Understanding the fundamentals of **image steganography** and **bitwise operations**.
* Working with **Python image processing** using Pillow.
* Designing both **CLI** and **GUI** applications in Python.
* Implementing **real-world encryption techniques**.
* Managing user inputs and file handling in both interface modes.
* Debugging and optimizing Python scripts for user-friendly applications.

**9. Conclusion:**

This **Steganography Tool** effectively demonstrates how Python can be used to create a functional and secure application for hiding messages in images. The support for both **CLI** and **GUI** makes it versatile and user-friendly. It serves as a practical example of how steganography can complement encryption to ensure data confidentiality and stealth. The project strengthens understanding in areas of image processing, security, and application development using Python.