CAPSTONE PROJECT

TEXT HIDING INSIDE IMAGES USING LSB STEGANOGRAPHY

PRESENTED BY:
STUDENT NAME:SANTHOSHAM VARSHINI
COLLEGE NAME:MALLAREDDY ENGINEERING COLLEGE FOR WOMEN
DEPARTMENT:INFORMATION TECHNOLOGY
EMAIL ID:SANTHOSHAMVARSHINI@GMAIL.COM
AICTE STUDENT ID:STU6641AF65EBCFC1715580773

edunet foundation

OUTLINE

- **Problem Statement** (Should not include solution)
- **System Development Approach** (Technology Used)
- **Algorithm & Deployment (Step by Step Procedure)**
- **Result**
- **Conclusion**
- **Future Scope(Optonal)**
- **References**

PROBLEM STATEMENT

- Secure communication often exposes the presence of hidden or encrypted data.
- Visible encryption can raise suspicion or invite interception.
- There's a need to hide messages in a way that doesn't attract attention.
- The solution should maintain the visual quality of the carrier image.
- The challenge is to embed and retrieve hidden text without detection.

SYSTEM APPROACH

1. System Requirements

- **Operating System :** Platform-independent (Linux, Windows, macOS via Google Colab or Jupyter Notebook)
- **Processor**: Intel/AMD dual-core or better
- **RAM**: Minimum 4 GB (8 GB recommended)
- **Storage**: At least 500 MB for image and file handling
- **Internet**: Required if using Google Colab or fetching external images

2. Libraries Required to Build the Model

Library	Purpose
NumPy	For pixel-level array manipulation
PIL (Pillow)	To handle image loading, editing, and saving
Matplotlib *(optional)*	To visualize images in notebooks
Google Colab Files	For uploading/downloading files in Colab
OpenCV *(optional)*	Advanced image handling (alternative to PIL)

ALGORITHM & DEPLOYMENT

Algorithm (Step-by-Step Logic)

1.Input Collection

- Take a **cover image** and **secret text message** as input.

2.Message Conversion

- Convert the text into its **binary representation**.
- Append a unique delimiter (e.g., 1111111111111110) to mark the end.

3.LSB Encoding

- Flatten the image array.
- Replace the **Least Significant Bit (LSB)** of each pixel with one bit of the message.
- Reshape and save the encoded image.

4.LSB Decoding

- Load the encoded image and extract LSBs from pixels.
- Stop at the delimiter, then convert the binary stream back to text.

Deployment Steps

1.Development Environment Setup

- Use **Google Colab** or **Jupyter Notebook**
- Install required libraries: numpy, PIL, matplotlib

2.User Interface (File Upload & Input)

- Allow users to **upload an image** and **enter text**
- Output: **Downloadable encoded image**

3.Testing & Validation

- Test with different image resolutions and message lengths.
- Check for any visual distortion or decoding errors.

4.Code Hosting

- Upload the notebook, images, and README to **GitHub**

5.Presentation

- Create a PPT summarizing:
  - Problem
  - System Design
  - Algorithm
  - Screenshots of Encoding/Decoding
  - Output and Conclusion

RESULT

- 1. Image Upload Section in Google Colab

```
# 📌 Required Libraries
from PIL import Image
import numpy as np
from google.colab import files
import io
```

```
# 📌 Upload Cover Image
print("📁 Please upload a cover image (PNG or JPG)...")
uploaded = files.upload()
img_path = list(uploaded.keys())[0]
```

- 2. Message Input and Binary Conversion (Encoding Part)

```
# 📌 Input secret message
secret_message = input("💬 Enter the secret message to hide: ")
```

3. LSB Encoding Code

```
# 📌 Embed message safely
for i in range(len(binary_msg)):
    flat_data[i] = (flat_data[i] & 254) | int(binary_msg[i])
```

```
# 📌 Reshape back and save encoded image
encoded_data = flat_data.reshape(data.shape)
encoded_img = Image.fromarray(encoded_data.astype('uint8'))
encoded_img.save("encoded_image.png")

# 📁 Download encoded image
print("✅ Message hidden successfully in 'encoded_image.png'")
files.download("encoded_image.png")
```

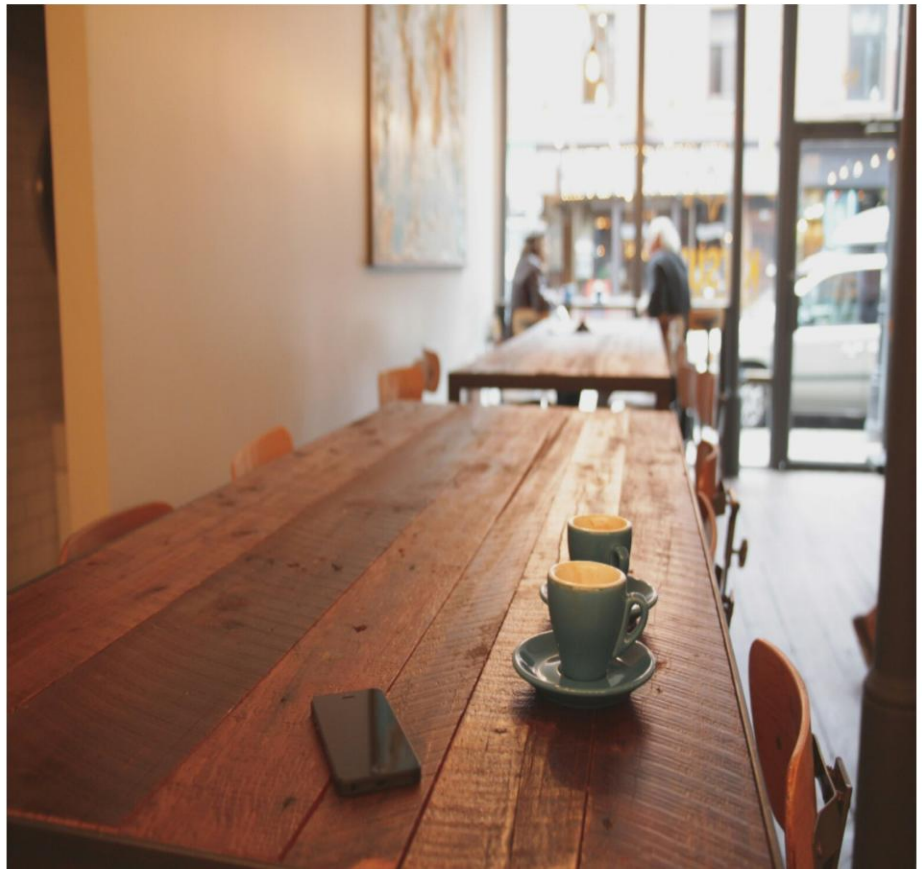
4.Encoding & output

```
# 📁 Upload the encoded image
print("📁 Upload the encoded image to extract the message...")
uploaded = files.upload()
encoded_img_path = list(uploaded.keys())[0]
```

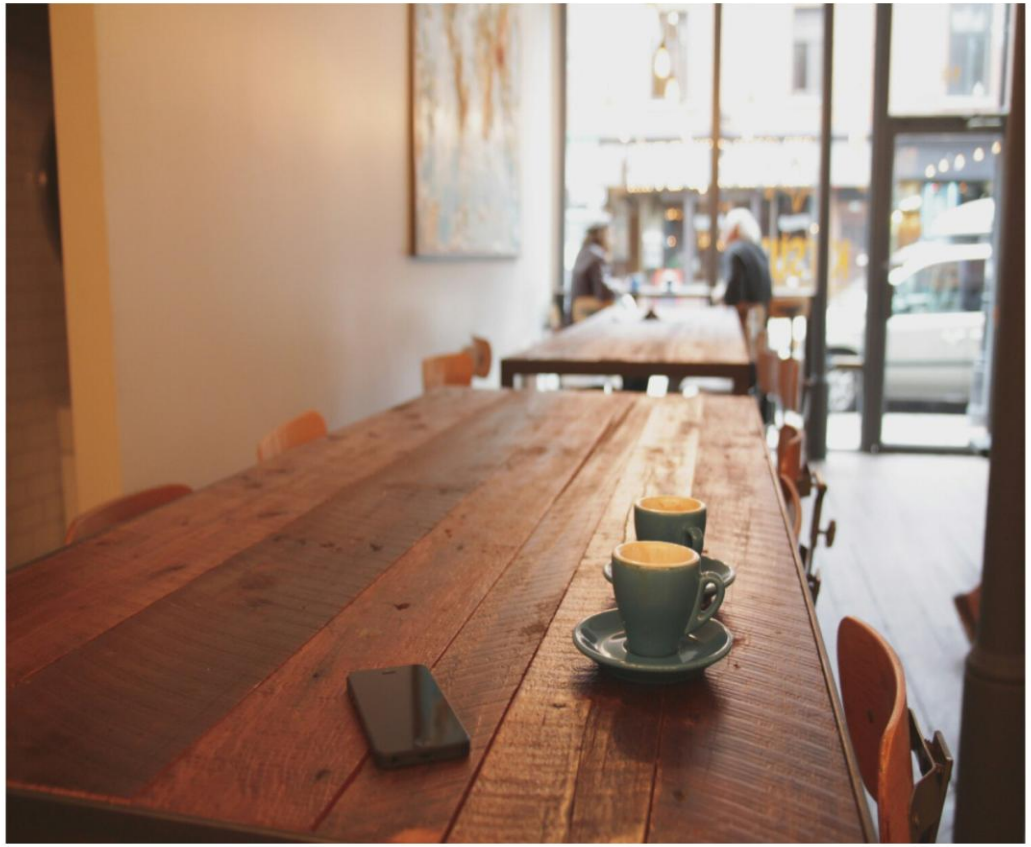
```
📁 Please upload a cover image (PNG or JPG)...
Choose Files luke-chess...unsplash.jpg
• luke-chesser-KR2mdHJ5qMg-unsplash.jpg(image/jpeg) - 702968 bytes, last modified: 6/20/2025 - 100% done
Saving luke-chesser-KR2mdHJ5qMg-unsplash.jpg to luke-chesser-KR2mdHJ5qMg-unsplash (1).jpg
💬 Enter the secret message to hide: hi IBM Skillsbuild glad to work with you!!
✅ Message hidden successfully in 'encoded_image.png'
```

```
📁 Upload the encoded image to extract the message...
Choose Files encoded_image (1).png
• encoded_image (1).png(image/png) - 5661280 bytes, last modified: 6/20/2025 - 100% done
Saving encoded_image (1).png to encoded_image (1).png
🔓 Hidden message extracted:
 hi Ibm skillsbuild,glad to work with you.!!
```

Original Image



Encrypted Image



Github Link:

https://github.com/Varshini1621/steganography-project

CONCLUSION

This project demonstrates how steganography can securely hide text inside images using the LSB technique.

The hidden message remains invisible and can only be retrieved using the correct decoding method.

The system works efficiently with lossless image formats like PNG using simple Python libraries.

It highlights the importance of data confidentiality in secure and covert communication.

FUTURE SCOPE(OPTIONAL)

- **Add Encryption**: Secure the hidden message with encryption (e.g., AES) before embedding it into the image.
- **Support for Other File Types**: Extend the system to hide files (PDF, ZIP) instead of plain text.
- **Multi-Channel Embedding**: Utilize all RGB channels more efficiently to increase hiding capacity.
- **GUI or Web App Interface**: Build a user-friendly interface using Tkinter or Flask for broader usability.
- **Use Deep Learning**: Explore deep steganography models for higher capacity and stronger robustness.

REFERENCES

•**Kharrazi, M., Sencar, H. T., & Memon, N. (2004).**
Image Steganography: Concepts and Practice
In: Proceedings of the 2nd International Workshop on Multimedia and Security
Link

•**Petitcolas, F. A., Anderson, R. J., & Kuhn, M. G. (1999).**
Information Hiding—A Survey
Proceedings of the IEEE, 87(7), 1062–1078.
DOI

•**Johnson, N. F., & Jajodia, S. (1998).**
Exploring Steganography: Seeing the Unseen
IEEE Computer, 31(2), 26–34.
Link