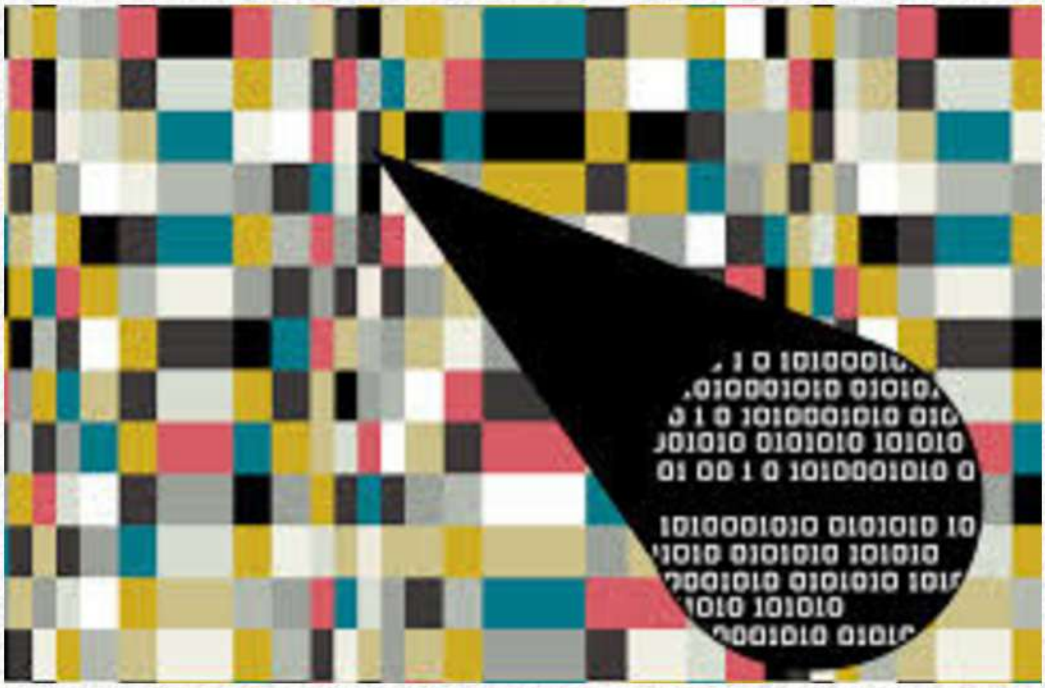
STEGANOGRAPHY



Name	ID
Ammar Saad Mohamed	20210588
Salma Ayman Abdelftah	20210411
Adam Mohamed Atia	20210133
Salma Mahmoud Fawzy	20210417
Ali Ibrahim Asaam	20210572
Ahmed Mohamed Abdalaziz	20210103

What is steganography?

Steganography is the practice of hiding information within other, non-secret, information or files, so that the hidden data is not apparent to someone who casually observes the carrier file. Unlike encryption, where the presence of a secret message is obvious (though the content is hidden), steganography conceals the fact that any message is present at all.

“steganography means hiding one peace of data within another”

Introduction to the Project

- Goal: To develop a user-friendly tool that embeds and extracts hidden messages in images using advanced steganographic methods.

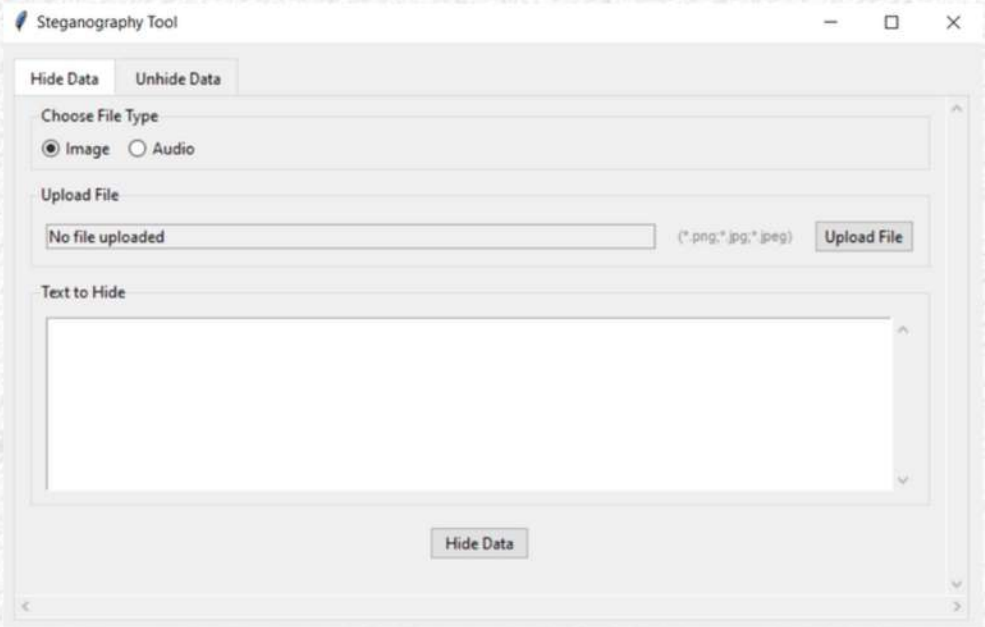
- Key Features:
  - Supports hiding text in both images and audio.
  - Uses Advanced Techniques
  - Simple and Easy-to-Use User Interface

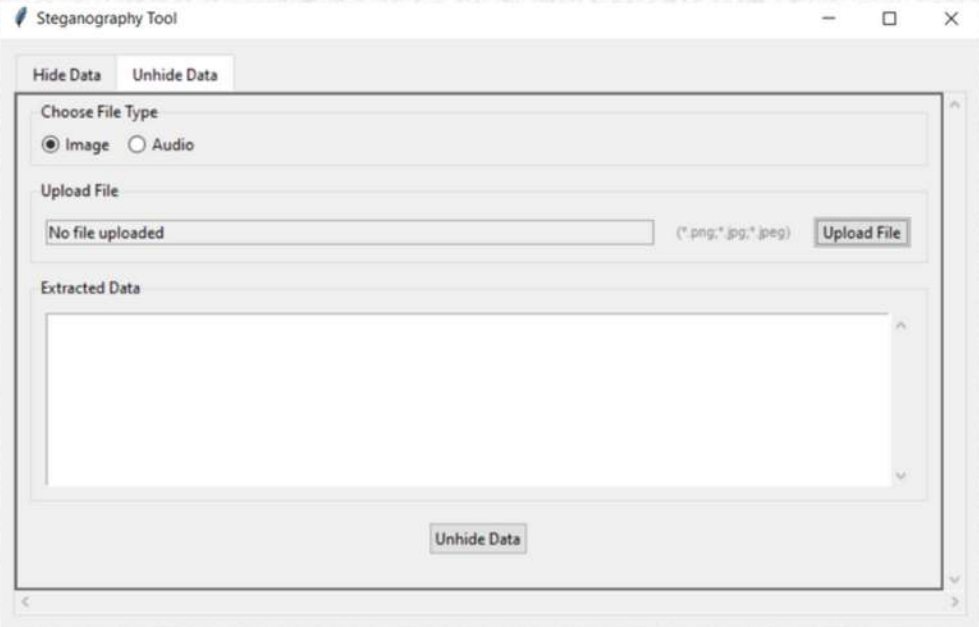
- Objective:
  - To develop a steganography tool that securely hides and extracts secret messages in images and audio.
  - Focus on balancing imperceptibility, data capacity, and usability.

User Interface

Graphical User Interface (GUI)

- First one was basic simple, The design is straightforward, making it user-friendly for tasks like hiding and revealing secret messages within media files.



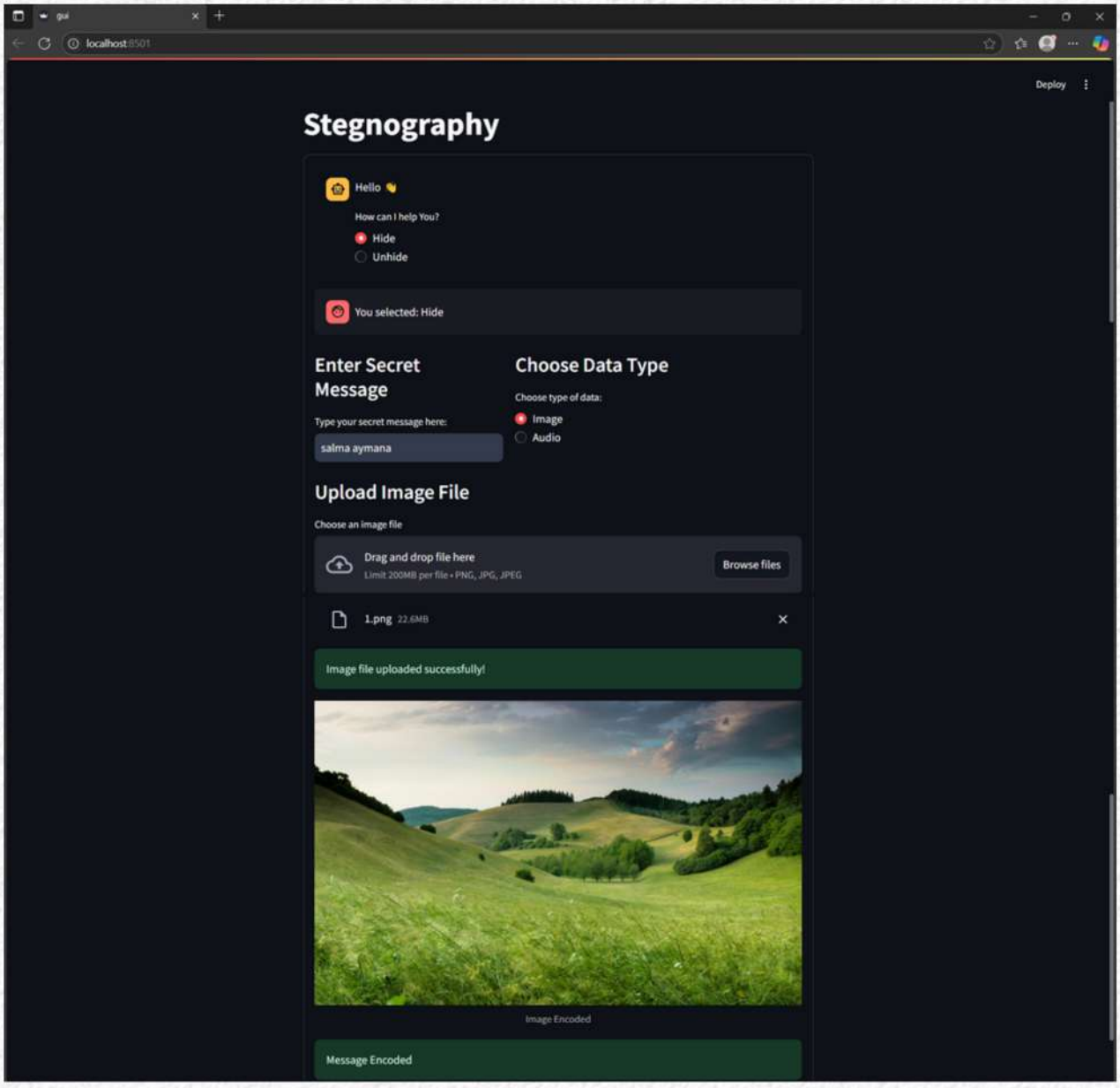


User Interface

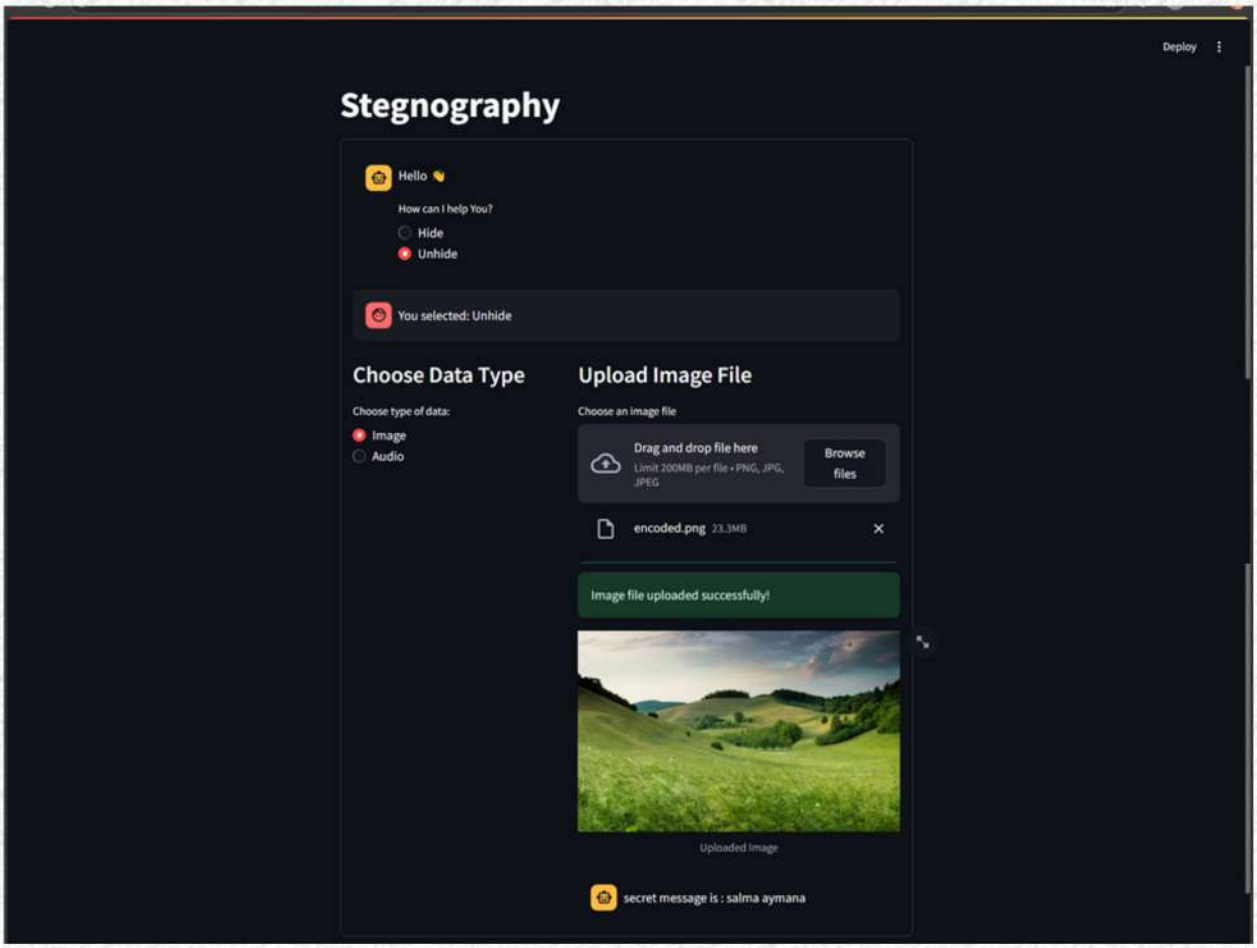
Streamlit (Application Version)

- ·Now we use Streamlit is clean, interactive, and user-friendly, focusing on simplicity and a guided experience for the user.

Hide



Unhide



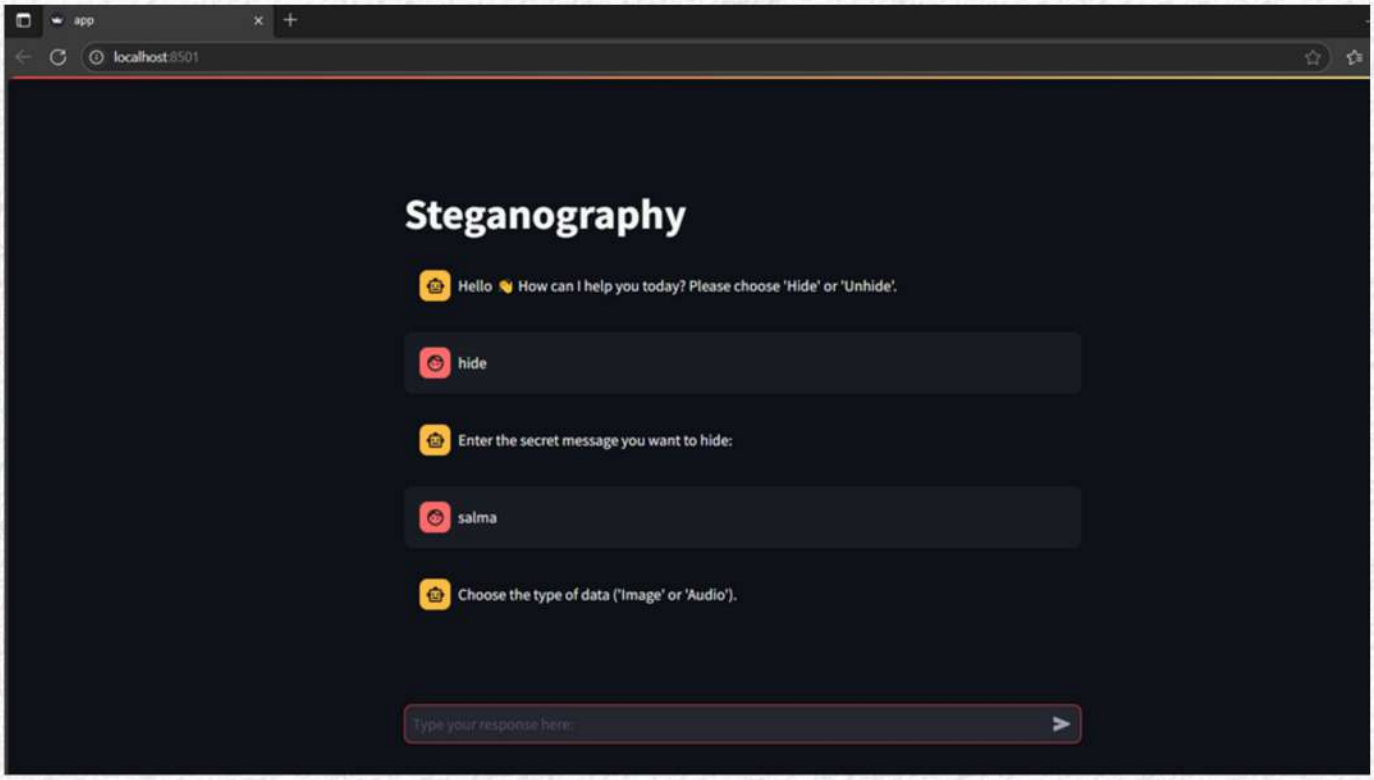
Why Use Streamlit for Your Steganography Project?

- easy to Use:
  - Simple Python scripts—no need for HTML, CSS, or JavaScript.
  - Quick to build interactive UIs with just a few lines of code.
  - Fast Prototyping:
  - Real-time updates—every change is instantly visible.
  - Ideal for testing and improving your steganography app quickly.
- Clean UI:
  - Modern and user-friendly interface out of the box.
  - Supports chat-based interactions, making the user experience smooth.
- File Handling:
  - Built-in support for file uploaders for images and audio.
  - Simplifies handling media files in steganography.

User Interface

Streamlit (Chatting Version)

- oproject is significantly improved with a chat-based interaction flow offers an interactive and conversational experience, guiding users through the steganography process (hiding or revealing secret messages) step-by-step



Challenge: Handling Media Transfer Between Bot and User:

- The current interface struggles with media file transfers (like sharing encoded images or audio files) between the bot and users.

Image steganography

LSB (Least Significant Bit):

- LSB (Least Significant Bit):
  - Embeds secret data by replacing the least significant bits of pixel values.
- Why LSB
  - Simple and efficient.
  - Minimal visual changes.
- Limitation:
  - Less secure for sensitive data due to vulnerability to steganalysis.
- Output:
  - It worked well without any problems, but we want to use advanced AI-based techniques.

Image steganography

PVD (Pixel Value Differencing):

- PVD (Pixel Value Differencing):
  - Hides data by analyzing the difference between adjacent pixel values.
- Why PVD
  - Embeds more data in high-contrast areas.
  - Maintains better image quality compared to LSB..
- Limitation:
  - Area with low contrast, the capcity is significantly reduced whic
    may restriced the overall playoad .
- Output:
  - It worked well without any problems, but we want to use
    advanced AI-based techniques.

Image steganography

Autoencoder:

- Autoencoders:
  - embed secret messages in the compressed latent space, allowing the decoder to reconstruct the image with the hidden message intact and undetectable.

Challenges of Using Autoencoders for Image Steganography

- 1.Visual Distortion in the Decoded Image:
  - The decoded image has visible artifacts and blurriness, indicating that the autoencoder might be losing fine details during the reconstruction process. This is typical when the latent space does not capture enough detail or when the model struggles to reconstruct high-frequency information.
- 2. Message Size Limitation
  - Initial attempts led to incorrect message extraction due to excessive compression.
  - Optimized message embedding and decoding layers for better accuracy.
- 3.Bit-Level Discrepancies:
  - The original message and decoded message differ at a few bit positions

Audio steganography

LSB (Least Significant Bit):

- LSB (Least Significant Bit):
  - Embeds secret data by replacing the least significant bits of audio samples.
- Why LSB
  - Quick and efficient for small-sized data.
  - Inaudible changes to audio quality.
- Limitation:
  - Can only hide small amounts of data without degrading quality.
  - Modifying LSBs can introduce noise in audio files.
- Output:
  - It worked well without any problems, but we want to use advanced AI-based techniques.

Audio steganography

Autoencoder:

- Autoencoders:
  - autoencoders are used to embed secret messages into audio signals by learning efficient representations that minimize detection. The encoder hides the message within the audio's latent features, while the decoder helps extract the hidden data accurately.

Challenges of Using Autoencoders for Audio Steganography

- 1. Feature Loss During Encoding and Decoding
  - The autoencoder compresses the MFCC features into a lower-dimensional space and then reconstructs them.
  - However, autoencoders are not perfect in reconstruction, meaning some details will be lost.
  - When embedding a message, even a small modification might get distorted when passing through the decoder.

- Error Type: Information Loss in Autoencoder Reconstruction This results in incorrect or garbled output when extracting the hidden message.

- 2. Incorrect Bit Embedding Precision
  - embed bits by modifying the first coefficient of the encoded MFCC features
  - However, after passing through the autoencoder’s nonlinear layers, this small change may be altered or lost.

- Error Type: Floating-Point Precision & Nonlinear Distortion This results in incorrect message extraction.

Future Scope

Image steganography

- Increase the size of the latent space to capture more details.
- Implement skip connections to retain high-frequency information.
- Fine-tune the encoder-decoder depth for better feature extraction.
- Expand latent vector size to accommodate both image and message information.
- Reduce compression level to balance embedding capacity and image quality.
- Use a hierarchical embedding approach for critical message bits.

Audio steganography

- Use More Robust Embedding
- Instead of small numerical changes, use redundant encoding (error correction codes) to improve robustness.
- Train the Autoencoder for Steganography, Not Just Compression
- Train the model explicitly to preserve small changes in MFCCs by adding a custom loss function.
- Use Spectrogram Instead of MFCC for Audio Reconstruction
- Instead of MFCCs, use Mel spectrograms with librosa.feature.melspectrogram() to make the process more reversible.

Conclusion

- This follow-up report highlights the ongoing development of the steganography project, focusing on integrating autoencoders and enhancing the chat-based user interface. Significant progress has been made in embedding and extracting secret messages with improved security and user experience. However, challenges such as information loss during reconstruction and precision in message embedding remain areas for further optimization. Moving forward, exploring more robust embedding techniques, training autoencoders specifically for steganography, and adopting spectrogram-based approaches for audio steganography could significantly enhance reliability and performance. Similarly, expanding the latent space and fine-tuning encoder-decoder architectures can improve image steganography outcomes. Overall, the project demonstrates promising potential and sets a clear direction for continued enhancements.