*Project on Enhanced Encryption Before LSB Capacity vs Quality Trade-Off (PSNR/SSIM)*

*What Your Final Project Will Do*

*1. User inputs a secret message.*

*2. Message is encrypted using AES (or XOR for simplicity).*

*3. Encrypted message is embedded into the LSBs of a grayscale image.*

*4. You vary the message length or embedding capacity (e.g., partial, full, over-capacity).*

*5. For each embedded image, you:*

*Compute PSNR (Peak Signal-to-Noise Ratio) and*

*SSIM (Structural Similarity Index)*

*Compare image quality and generate a graph or table of the trade-off.*

***Code structure***

***1. encryption.py***

*Handles encryption and decryption of the secret message.*

* *AES encryption/decryption (or XOR for simplicity)*
* *Key generation and handling*
* *Padding/unpadding if needed*

***2. lsb\_embedder.py***

*Embeds and extracts encrypted messages using LSB steganography.*

* *Load grayscale image*
* *Embed encrypted message into LSBs*
* *Extract message from LSBs*

***3. quality\_metrics.py***

*Computes image quality metrics.*

* *PSNR calculation*
* *SSIM calculation*
* *Optionally: MSE (Mean Squared Error)*

***4. main.py***

*The main driver script that ties everything together.*

* *Takes user input*
* *Calls encryption → embedding → quality analysis*
* *Varies message length/capacity*
* *Generates graphs/tables for trade-off analysis*

***5. Optional: visualizer.py***

*If you want to keep plotting separate, this file can generate:*

* *Graphs of PSNR/SSIM vs message length*
* *Tables comparing original vs stego images*