Secure File Sharing System Report – Task 3

Intern Name: Lokesh Indala
Internship Track: Cyber Security
Task Title: Secure File Sharing System
Date: July 2025
Tools Used: Python, Flask, PyCryptodome

Objective :

The goal of this task was to build a secure file sharing system that supports ;

- File upload by users
- Encryption of uploaded files
- Secure download of decrypted files

This simulates secure file transfer practices used in cybersecurity and enterprise systems.

Tools and Technologies :

Tool/Library	**Purpose**
Python	Programming language
Flask	Web framework for building upload/download APIs
PyCryptodome	Encryption library for AES implementation
HTML (templates)	Frontend for file upload form
VS Code	Code editor
GitHub	Version control and code publishing

Encryption Method Used :

- Algorithm : AES (Advanced Encryption Standard)
- Mode : CBC (Cipher Block Chaining)
- Key Size: 128-bit (16 bytes)
- Padding : (PKCS7)

Encryption is handled using a randomly generated AES key IV, with secure storage and retrieval of data. The decrypted file restores the original content accurately.

Features Implemented :

Feature	Status
File upload via web form	✅ Done
File encryption using AES	✅ Done
Encrypted file storage	✅ Done
Download + decryption route	✅ Done
Auto-creation of folder paths	✅ Done

Folder Structure :

```
secure_file_share/
├── app.py                  # Flask application
├── templates/
│   └── upload.html         # HTML form
├── uploads/                # Raw uploaded files
├── encrypted/              # AES-encrypted files
└── decrypted/              # Final decrypted files
```

Testing Performed :

1. Created a file **sample.txt** with test content
2. Uploaded via the form at **http://127.0.0.1:5000/**
3. Confirmed encryption and saved output to **/encrypted/**
4. Downloaded and decrypted file via **/download/sample.txt.enc**
5. Verified content matched original input

Security Considerations :

- Used AES in CBC mode to ensure block-level encryption
- Randomly generated IV per file for added security
- No plaintext file exposed outside secure folders
- Decryption only happens on-demand during download

Outcome :

Successfully built a secure file upload and download system using Python and Flask, with end-to-end AES encryption. The project simulates real-world secure data transfer used in government, finance, and cloud environments.