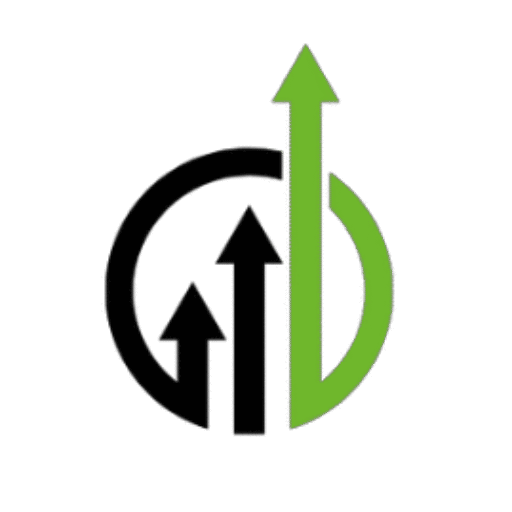
Secure File Sharing System



**Program:** Future Interns Cyber Security Program

**Task 2:** Secure File Sharing System

**Name:** Ilegbemi Damilola Olumide

**Date:** September 2025

**Tools used:** Python Flask, PyCryptodome (AES encryption) , HTML/CSS, venv.

## Introduction

In this task, I developed a Secure File Sharing System using Python Flask as the backend framework and AES encryption for data protection. The system allows users to upload files, which are encrypted before storage, and download files, which are decrypted before delivery. This project simulates a real-world solution for secure data sharing in sensitive industries such as healthcare, legal, and corporate environments.

## Tools and Technologies Used

- Python Flask : for building the web application  
- PyCryptodome (AES encryption) : for encrypting and decrypting files  
- HTML / CSS : for creating a simple user interface  
- Virtual Environment (venv) : for project isolation  
- Web Browser (localhost) : to access and test the application

## Implementation Steps

1. Environment Setup

• Installed Flask and PyCryptodome.  
 • Created project folders: templates/, uploads/.  
 • Generated a secure AES key using Python and stored it as an environment variable.

2. Application Development

• Wrote app.py with Flask routes for uploading & encrypting files, listing available files, and downloading & decrypting files.  
 • Created index.html in templates/ for the user interface.

3. Running the App

• Activated virtual environment and started Flask server.  
 • Accessed the app at: http://127.0.0.1:5000.

## Identified Security Features

1. AES Encryption (Advanced Encryption Standard)  
 • Files are encrypted on upload using AES in CFB mode.  
 • Prevents unauthorized access to file contents at rest.  
  
2. Decryption on Download  
 • Files are decrypted only when downloaded, ensuring end-to-end confidentiality.  
  
3. Key Management  
 • The AES key is securely generated and stored as an environment variable.  
 • Ensures that keys are not hardcoded in the source code.

## Security Overview

This system protects sensitive files through encryption, ensuring confidentiality and integrity. If attackers gain access to stored files, they will still be unreadable without the AES key. This reflects real-world secure file sharing practices used by organizations.

## Conclusion

This project provided hands-on experience in:  
 • Flask web development  
 • Implementing encryption with AES  
 • Secure file handling and basic cryptographic key management  
  
By building this system, I strengthened my knowledge in data security, secure software design, and SOC-relevant skills such as ensuring confidentiality during file transfer and storage.